

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

**до виконання розрахунково-графічної роботи
з дисципліни «Системи технічного зору»**

для студентів спеціальностей
7.05090201, 8.05090201 «Радіоелектронні апарати та засоби»

Рекомендовано
Вченою радою факультету
електроніки

Протокол № 09/2015
від 28 вересня 2015 р.

Рекомендовано
кафедрою конструювання електронно-
обчислювальної апаратури

Протокол № 12 від 09 вересня 2015 р.
Завідувач кафедри

_____ О.М. Лисенко

Методичні вказівки до виконання розрахунково-графічної роботи з дисципліни «Системи технічного зору» для студентів спеціальностей 7.05090201, 8.05090201 «Радіoeлектронні апарати та засоби» / Уклад. Варфоломєєв А.Ю., Дзюба В.Г. – К.: НТУУ «КПІ», 2015. – 41 с.

Е л е к т р о н н е н а в ч а л ь н е в и д а н н я

Методичні вказівки

до виконання розрахунково-графічної роботи
з дисципліни «Системи технічного зору»
для студентів спеціальностей
7.05090201, 8.05090201 «Радіoeлектронні апарати та засоби»

Укладачі: Варфоломєєв Антон Юрійович, к.т.н. ст. викладач
Дзюба Віталій Георгійович, к.т.н.

Відповідальний редактор: Лисенко Олександр Миколайович
докт. техн. наук, проф.

За редакцією укладачів

ЗМІСТ

ВСТУП	4
ПРОЦЕДУРА РОЗПІЗНАВАННЯ ІНДЕКСУ НА КОНВЕРТАХ	5
Перетворення кольорового зображення у півтонове	7
Бінаризація півтонового зображення з автоматичним визначенням порогу	9
Виділення компонент зв'язності	12
Визначення маркерів індексу та фільтрація помилкових об'єктів	13
Знаходження початкового та кінцевого маркеру	15
Визначення кута нахилу	16
Впорядкування розташування маркерів	16
Поворот зображення	19
Виділення зони індексу	23
Сегментування зони індексу	24
Розпізнавання зони індексу	25
Простий нейрон	25
Функції активації	26
Нейрон з векторним входом	27
Архітектура нейронних мереж	28
Одношарова мережа	28
Багатошарові мережі	30
Формування навчальної вибірки та навчання мережі	32
Формування архітектури мережі	33
Навчання нейронної мережі	34
Розпізнавання цифр індексу	35
ЗАВДАННЯ	38
ВИСНОВКИ	39
ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ	40
ЛІТЕРАТУРА	41

ВСТУП

Сучасні досягнення в області електроніки та обчислювальної техніки створюють сприятливу можливість використання звичайних комп'ютерів при вирішенні складних інженерних та наукових задач. В свою чергу рівень розвитку прикладних областей штучного інтелекту, таких як нейронні мережі, нечітка логіка, генетичні і еволюційні алгоритми, інтелектуальні агенти дозволяє у поєднанні з новітніми методами і алгоритмами цифрової обробки сигналів підійти впритул до вирішення проблеми розпізнавання зображень.

Область комп'ютерного зору включає великий комплекс задач по автоматичному аналізу зображень, таких як розпізнавання об'єктів, осіб або рукописного тексту, обробка зображень в медицині, вилучення зображень з баз даних по їх текстовому опису або нарису від руки, контроль якості промислової продукції, виявлення і стеження за об'єктами, реконструкція поверхонь, організація зорового зворотного зв'язку при роботі керованих пристроїв, маніпуляторів або мобільних роботів в мінливому середовищі та інше. Спеціалізовані комп'ютерні системи аналізу зображень, що розробляються для вирішення таких задач, адекватно функціонують в певних умовах і чутливі до їх змін. Створення системи комп'ютерного зору з гнучкими можливостями, які подібні до зорової системи людини – складна наукова і практична задача.

Задача розпізнавання індексу на конвертах є прикладною і може знайти застосування для автоматизації процесу доставки листів.

Мета роботи: розробити алгоритм пошуку і розпізнавання індексу на конверті та реалізувати його у вигляді Simulink моделі системи MATLAB. Досягнення мети передбачає розв'язання таких задач:

- 1) перетворення кольорового зображення в півтонове;
- 2) бінарізація півтонового зображення з автоматичним визначенням порогу;
- 3) виділення компонент зв'язності;
- 4) визначення маркерів індексу та фільтрація помилкових об'єктів;
- 5) знаходження початкового та кінцевого маркеру;
- 6) визначення кута нахилу;
- 7) впорядкування розташування маркерів;
- 8) поворот зображення;
- 9) виділення зони індексу;
- 10) сегментування зони індексу;
- 11) розпізнавання зони індексу.

ПРОЦЕДУРА РОЗПІЗНАВАННЯ ІНДЕКСУ НА КОНВЕРТАХ

Для реалізації процедури розпізнавання індексу на конвертах у вигляді Simulink моделі потрібно створити в системі MATLAB нову модель. Для цього у головному меню програми обираємо послідовність таких його елементів: File → New → Model (рис. 1).

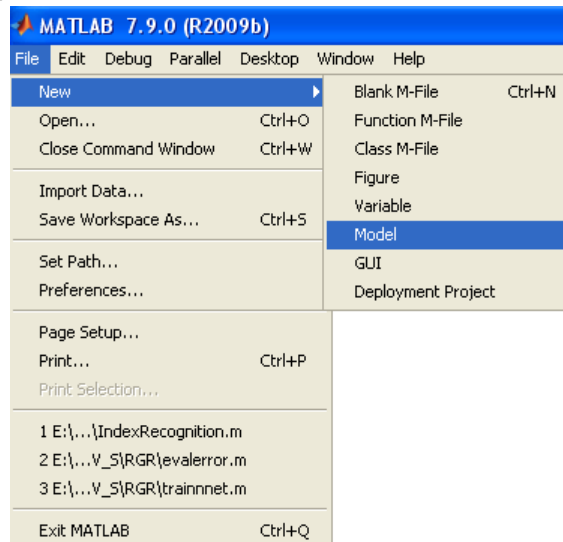


Рисунок 1 – Створення нової моделі

В результаті з'являється вікно нової моделі, що має назву «untitled» (рис. 2). Збережемо модель під назвою «EnvelopeRecognition». Зверніть увагу, що назви як моделі, так і її елементів (блоків) вкрай бажано давати латиницею, оскільки при невиконанні цієї умови Simulink може видавати помилки, а збережена модель може після її закриття взагалі не відкритися.

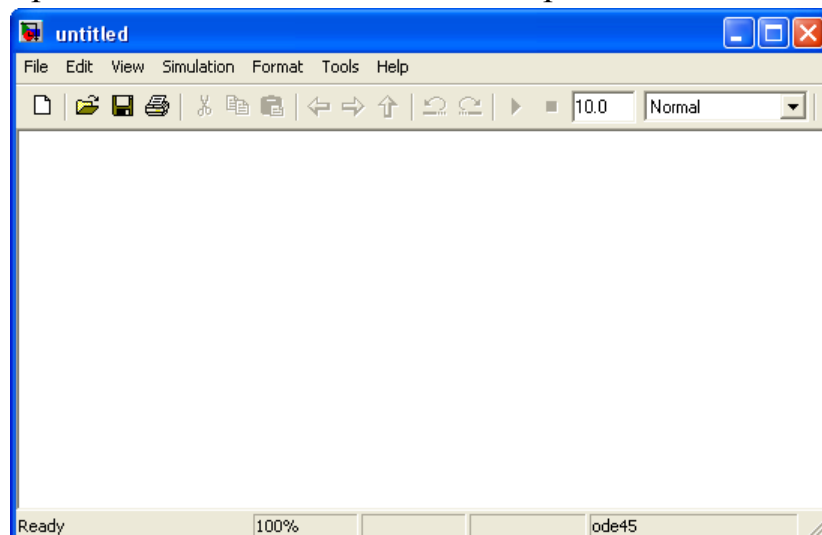


Рисунок 2 – Вікно новоствореної моделі

Перше, що має вміти виконувати розроблювана модель – це зчитування та відображення зображень. Для того, щоб реалізувати в ній зазначений

функції, необхідно використати відповідні блоки, які можна знайти, перейшовши до *бібліотеки блоків*, для чого в головному меню вікна перейти до View → Library Browser. У вікні, що відкриється слід обрати групи блоків Sinks (приймачі даних) та Sources (джерела даних), які є складовими Computer Vision System Toolbox (рис. 3).

В групі Sources бібліотеки блоків оберемо та перетягнемо у вікно моделі EnvelopeRecognition елемент Image From File, а з групи Sinks – елемент To Video Display. Далі за допомогою курсору з'єднаємо вихід елементу Image From File та вхід блоку To Video Display.

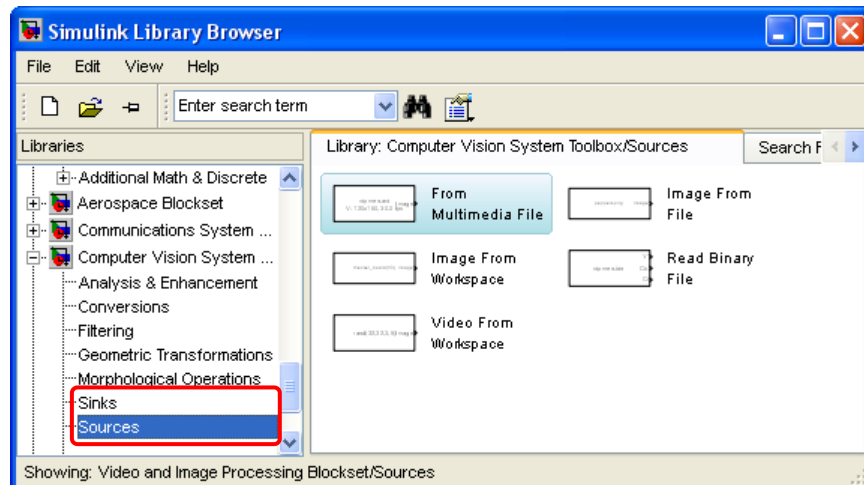


Рисунок 3 – Вікно бібліотеки блоків

Для налаштування моделі натисніть Simulation → Configuration Parameters... У відкритому вікні виберемо категорію «Solver» та встановимо такі параметри в групі Solver Options (див. рис. 4):

- ✓ Type → Fixed-step;
- ✓ Solver → discrete (no continuous states).

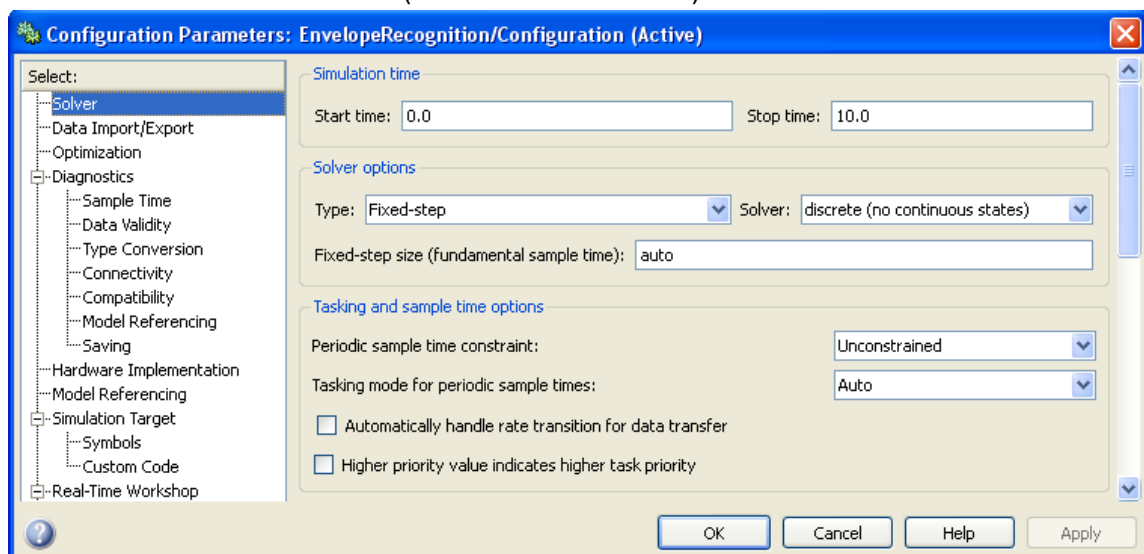

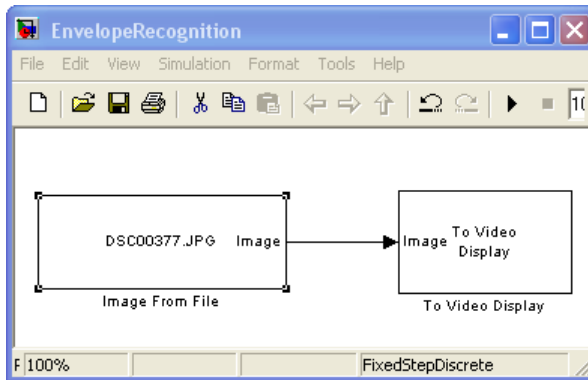
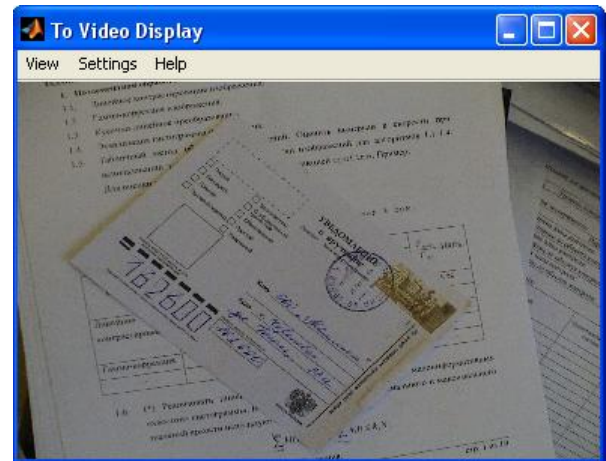


Рисунок 4 – Вікно параметрів моделювання

Запустити на виконання модель можна, якщо в головному меню вибрати Simulation → Start або натиснути сполучення клавіш Ctrl + T чи на піктограму  з панелі інструментів. Після цього повинно відобразитись зображення «peppers.png», яке встановлено по замовчуванню. Змінимо його на зображення «DSC00377.jpg», що знаходиться в папці «Test_Images». Робоча модель та результат її роботи має вигляд такий, як показано на рисунку 5.



а)



б)

Рисунок 5 – Simulink-модель – а та результат її роботи – б

Нижче детальніше розглянемо алгоритм розпізнавання індексу на конвертах та створимо повну модель, що його реалізує.

Перетворення кольорового зображення у півтонове

Розпізнавання індексу на конвертах базується на знаходженні так званих маркерів, які є вказівниками місцезнаходження зони індексу (див. рис. 6). Присутність інформації про колір при розпізнаванні є зайвою, тому подальша робота здійснюватиметься із півтоновим зображеннями.

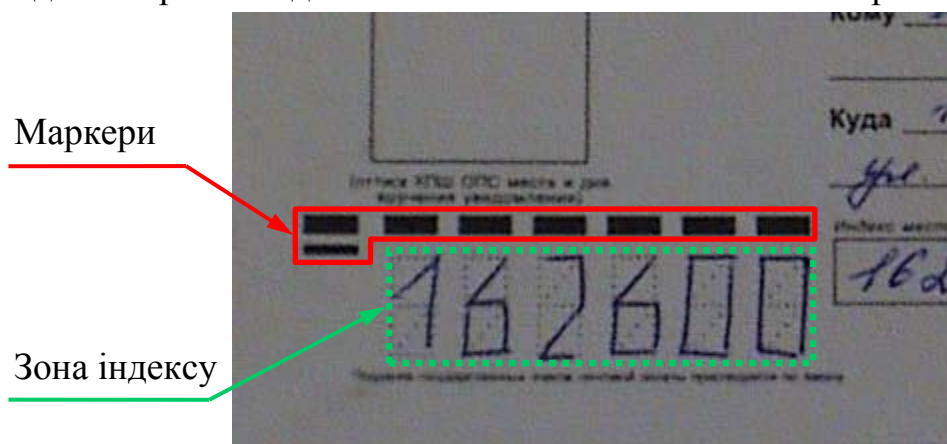


Рисунок 6 – Ключові об'єкти при розпізнаванні індексу на конвертах

Таким чином наступною задачею, яку слід вирішити – перетворити вхідне зображення у півтонове.

З метою досягнення найбільшої природності перетворення кольорового RGB зображення у півтонове, його доцільно здійснюється з урахуванням чутливості ока людини до різних кольорів. На рис. 7 наводиться нормована крива чутливості ока до електромагнітного випромінювання з різною довжиною хвилі, так звана крива видності.

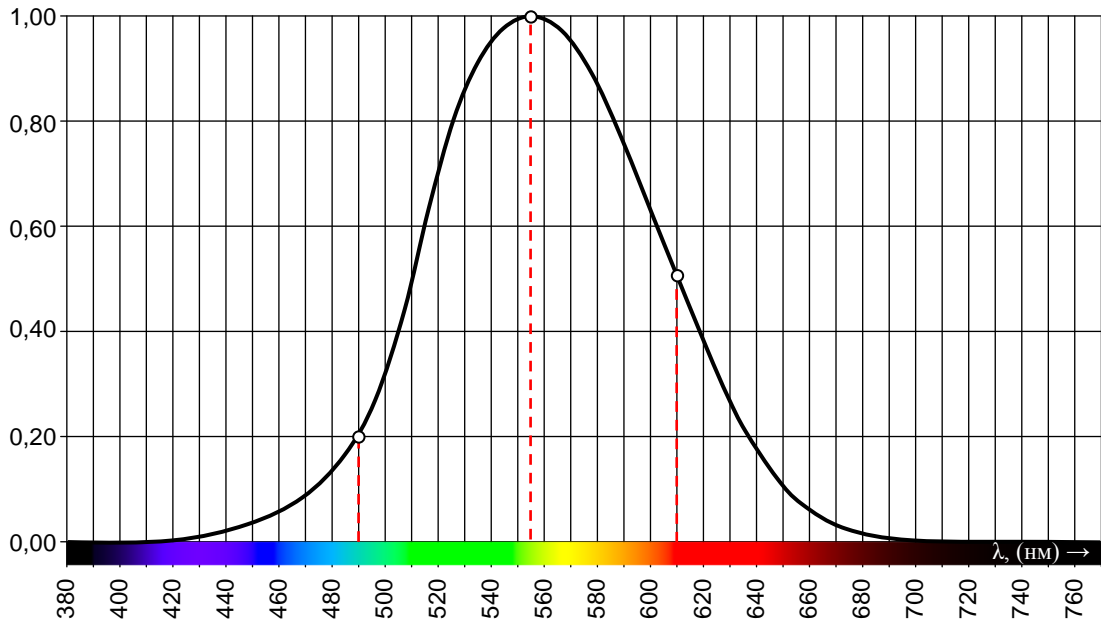


Рисунок 7 – Крива нормованої спектральної чутливості ока людини

З даної кривої видно, що око людини має найвищу чутливість до зеленого кольору, її значення приймається рівною 1. До червоного кольору чутливість ока менша приблизно в 2 рази, і її відносне значення становить 0,5. Найменшою чутливістю є до синього кольору, яка відносно чутливості до зеленого менша у 5 разів, і складає 0,2.

Півтонове зображення – значення яскравості Y в кожній точці даного кольорового зображення. Для отримання Y , що відповідає хроматичній чутливості ока необхідно виконати додавання трьох основних кольорів з урахуванням отриманих вище співвідношень, та виконати нормування, тобто:

$$Y = \frac{0,5R + 1G + 0,2B}{0,5 + 1 + 0,2} = \frac{5}{17}R + \frac{10}{17}G + \frac{2}{17}B \approx 0,3R + 0,59G + 0,11B$$

Більш точна формула перетворення кольорового зображення у півтонове є наступною:

$$Y = 0,299R + 0,587G + 0,114B$$

Для виконання такого перетворення скористаємось блоком «Color Space Conversion», який знаходиться в групі «Conversions» бібліотеки

блоків. Для цього блоку встановимо значення «R'G'B' to intensity» для параметру «Conversion» (рис. 8).

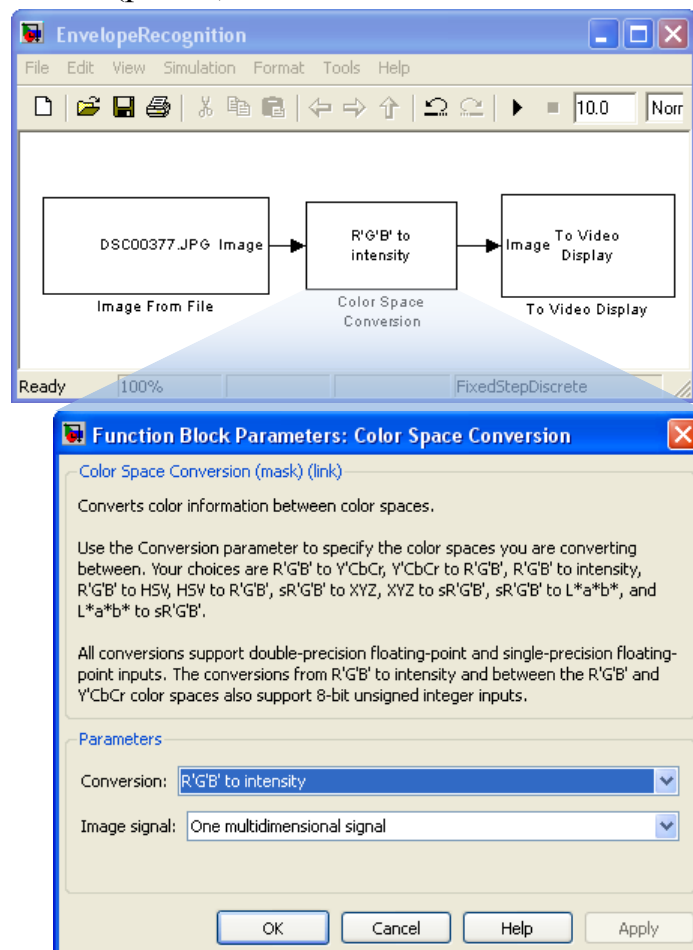


Рисунок 8 – Налаштування параметрів блоку перетворення кольорових просторів

Якщо знову запустити модель, то можна впевнитися, що на виході присутнє півтонове зображення.

Бінаризація півтонового зображення з автоматичним визначенням порогу

Для коректного перетворення півтонового зображення на двійкове необхідно правильно визначити поріг. Одним з найпоширеніших методів автоматичного його визначення є метод Отсу, ефективність якого доведена багаторічною практикою [3]. Даний метод ґрунтується на аналізі нормалізованих гістограм, що представляються у вигляді дискретної функції розподілу ймовірностей:

$$p_r(r_q) = \frac{n_q}{n}, \quad q = 0, 1, 2, \dots, (L - 1)$$

де n – це загальне число пікселів зображення, n_q позначає число пікселів, яскравість яких дорівнює величині r_q , а L – число різних рівнів яскравості зображення. Нехай тепер поріг k обраний так, що C_0 позначає множину

пікселів з рівнями яскравості з $[0, 1, \dots, k-1]$, а C_0 – множина пікселів з рівнями яскравості з $[k, k+1, \dots, L-1]$. Метод Отсу полягає у виборі порогу k таким чином, щоб він максимізував дисперсію між класами, яка дорівнює σ_B^2 :

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2,$$

де

$$\begin{aligned} \omega_0 &= \sum_{q=0}^{k-1} p_r(r_q), \quad \omega_1 = \sum_{q=k}^{L-1} p_r(r_q), \\ \mu_0 &= \sum_{q=0}^{k-1} \frac{qp_r(r_q)}{\omega_0}, \quad \mu_1 = \sum_{q=k}^{L-1} \frac{qp_r(r_q)}{\omega_1}, \quad \mu_T = \sum_{q=k}^{L-1} \frac{qp_r(r_q)}{\omega_0}. \end{aligned}$$

Отже, функція автоматичного визначення порогу приймає вхідне зображення, обчислює його гістограму, а потім знаходить порогову величину k , яка максимізує σ_B^2 . Для подальшої бінарізації зображення виконується порогове перетворення наступним чином:

$$g(x, y) = \begin{cases} 1, & \text{якщо } f(x, y) > k \\ 0, & \text{якщо } f(x, y) \leq k \end{cases}$$

де $f(y, x)$ – вихідне зображення, а $g(y, x)$ – бінарізоване.

Блок «Autothreshold», який знаходиться в групі «Conversions» в бібліотеці блоків, повністю реалізує бінарізацію півтонового зображення з автоматичним визначенням порогу за методом Отсу. Додавши його до моделі, отримаємо результат показаний на рисунку 9.

Оскільки для подальшого аналізу будуть використовуватись операції бінарної морфології, то, очевидно, отримане зображення має не досить зручне представлення, так як об'єкти інтересу (маркери індексу) є чорними (мають нульові значення яскравості, а отже морфологічні операції з ними виконувати не вдасться). Виправити цю ситуацію можна за допомогою блоку «Image Complement», який перетворює вихідне зображення в негативне і розташовуватиметься в моделі одразу за блоком «Autothreshold».

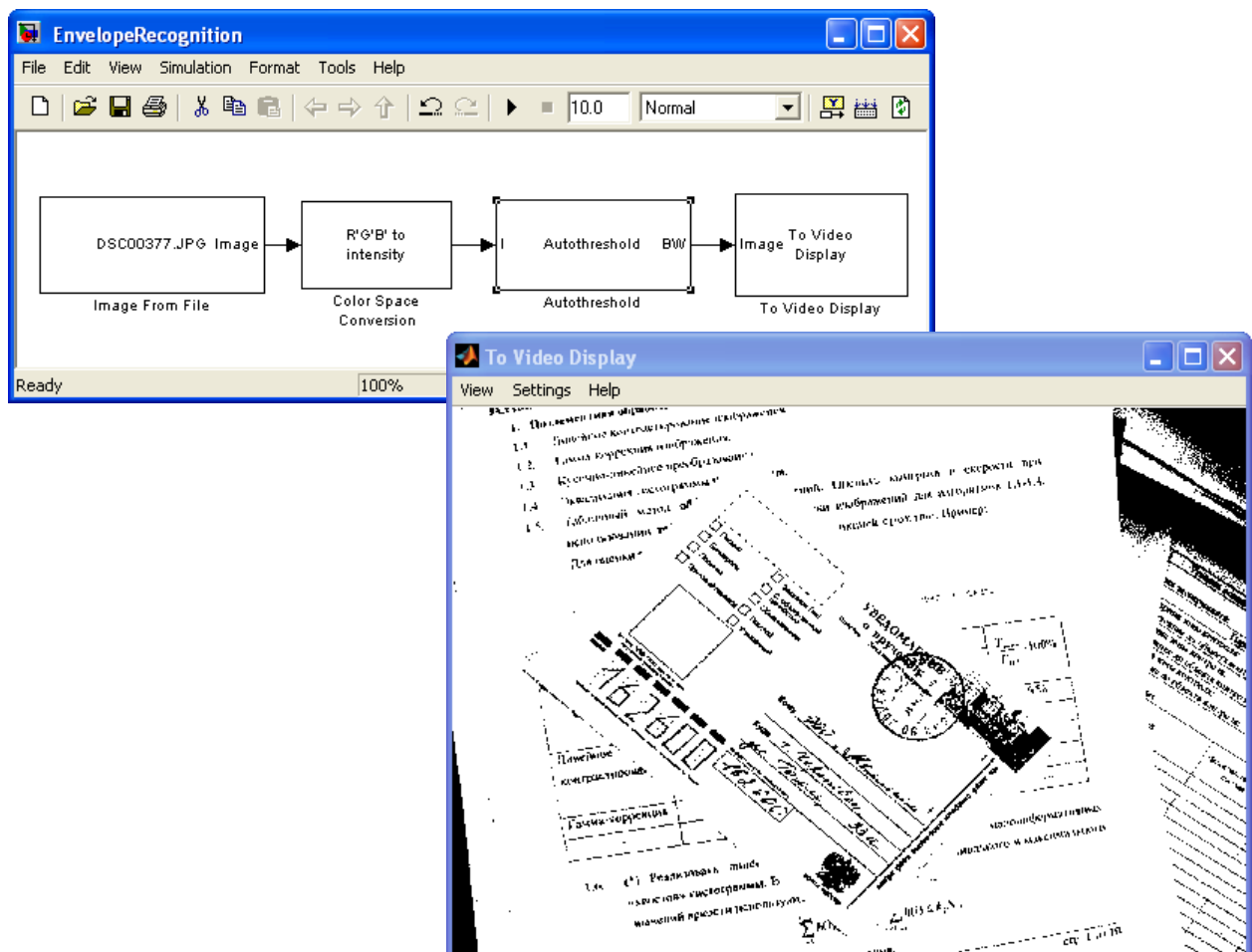


Рисунок 9 – Приклад роботи моделі, що здійснює бінарізацію півтонового зображення

Наступним етапом є пошук маркерів індексів на бінарному зображенні. Даний етап передбачає вирішення наступних задач:

- 1) виділення компонент зв'язності;
- 2) фільтрація помилкових об'єктів;
- 3) знаходження початкового та кінцевого маркеру;
- 4) визначення кута нахилу;
- 5) впорядкування розташування маркерів.

Для цього включимо в модель додаткову підсистему «Detect Markers» шляхом перетаскування блоку «Subsystem» з групи «Commonly Used Blocks». Вхідний її порт прийматиме бінарне зображення (назвемо його «BWin»), а порт на виході видаватиме статистичну інформацію по знайденим маркерам (відповідно дамо назву «STATS») див. рис. 10.

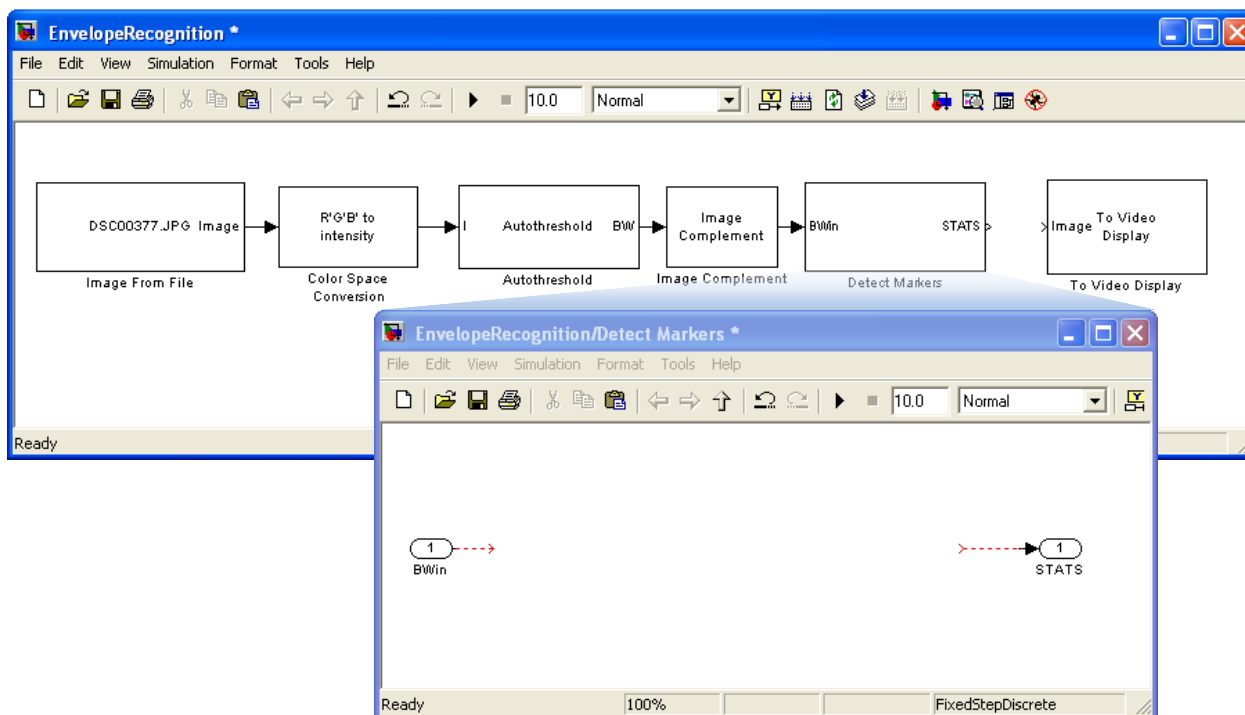


Рисунок 10 – Включення до моделі блоку підсистеми «Detect Markers»

Виділення компонент зв'язності

Візуальний аналіз зовнішнього вигляду маркерів індексу дозволяє формалізувати їх опис: маркер індексу є прямокутною областю суміжних пікселів, яскравість яких дорівнює одиниці у випадку бінарного зображення.

На цій концепції будується подальший алгоритм пошуку маркерів, перший крок якого полягає у виділенні компонент зв'язності. Нагадаємо, що є компонентою зв'язності.

Піксель p з координатами (x, y) має два горизонтальних і два вертикальні сусіди з координатами $(x + 1, y)$, $(x - 1, y)$, $(x, y + 1)$ та $(x, y - 1)$. Ця четвірка сусідів p позначається як $N_4(p)$. Чотири діагональні сусіди p мають координати $(x + 1, y + 1)$, $(x + 1, y - 1)$, $(x - 1, y + 1)$ та $(x - 1, y - 1)$. Ця четвірка позначається як $N_D(p)$. Об'єднання $N_4(p)$ із $N_D(p)$ визначає вісімку сусідів пікселя p , яка позначається $N_8(p)$. Два пікселі p і q називаються 4-суміжними (зв'язними), якщо $q \in N_4(p)$. Аналогічно, пікселі p і q називаються 8-суміжними (зв'язними), якщо $q \in N_8(p)$.

Шляхом між пікселями p_1 і p_n називається послідовність пікселів p_1, p_2, \dots, p_n , така, що p_k є суміжним для p_{k+1} при $k=1, 2, \dots, n-1$. Шлях може бути 4-зв'язним або 8-зв'язним, в залежності від виду суміжності.

Два пікселі p і q , що мають значення «1» називаються 4-зв'язними, якщо існує 4-зв'язний шлях між ними, який цілком складається з пікселів, що мають значення «1». Пікселі p та q називаються 8-зв'язними, якщо між ними існує 8-зв'язний шлях. Для будь-якого пікселя p множина всіх пов'язаних з

ним пікселів одиничних пікселів називається компонентою зв'язності, що містить p .

На практиці виділення компонент зв'язності в двійковому зображенні займає центральне місце в багатьох прикладних завданнях аналізу зображень. Нехай Y – деяка компонента множини A , що міститься на зображенні, і припустимо, що відома точка $p \in Y$. Тоді всі елементи компоненти Y можуть бути отримані за допомогою наступного рекурентного співвідношення:

$$X_k = (X_{k-1} \oplus B) \cap A, \quad k = 1, 2, 3, \dots$$

де $X_0 = p$; \oplus – операція дилатації; B – відповідний примітив, що задає тип зв'язності (4- або 8-зв'язність). Якщо $X_k = X_{k-1}$, то це говорить про збіжність алгоритму, і можна прийняти $Y = X_k$.

Виділення компонент зв'язності реалізує блок «Label». При його використанні на виході достатньо мати матрицю розмітки, інформація щодо кількості виділених компонент зв'язності є непотрібною, тому параметр «Output» цього блоку ініціалізується значенням «Label matrix». Для перетворення в формат плаваючої крапки з подвійною точністю елементів розмічувальної матриці додатково застосовуємо блок «Data Type Conversion» (рис. 11).

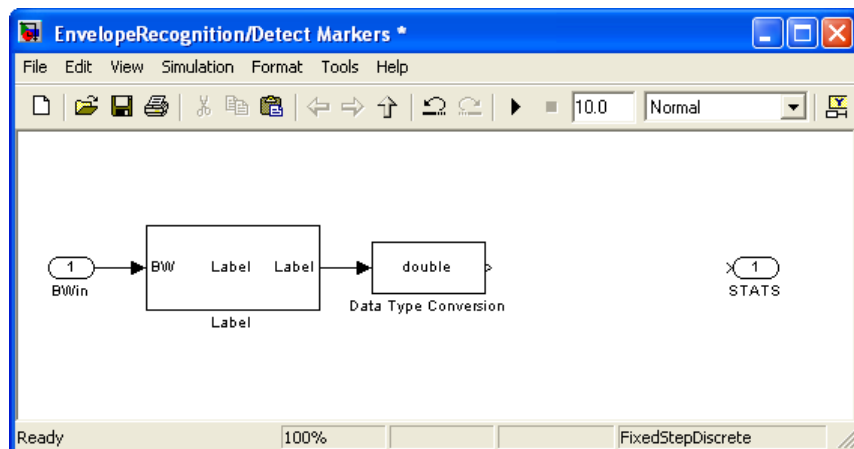


Рисунок 11 – Виділення компонент зв'язності в блоці «Detect Markers»

Визначення маркерів індексу та фільтрація помилкових об'єктів

Як вже зазначалося вище, маркер індексу є прямокутною областю з певним відношенням сторін, отже, для фільтрації маркерів використовуватимемо певні критерії, що обчислюватимуться на основі геометричних характеристик областей зв'язності.

Визначення характеристик областей зв'язності в системі MATLAB здійснюється за допомогою функції **regionprops**. В задачі, що розглядається цікавими є наступні величини:

- **'MajorAxisLength'** – довжина (в пікселях) великої вісі еліпсоїда, який має ідентичні другі центральні моменти, що й область зв'язності (рис. 12);
- **'MinorAxisLength'** – довжина (в пікселях) малої вісі еліпсоїда, який має ідентичні другі центральні моменти, що й область зв'язності;
- **'FilledArea'** – кількість пікселів двійкового зображення з розмірами, що аналогічні розмірам обмежувального прямокутника;
- **'ConvexArea'** – кількість пікселів двійкового зображення випуклої оболонки, що апроксимує область зв'язності;
- **'Centroid'** – центр мас області зв'язності;
- **'Orientation'** – кут між віссю абсцис та великою віссю еліпсоїда.

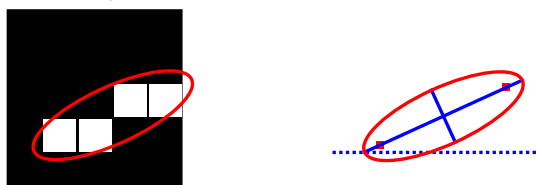


Рисунок 12 – Апроксимація області зв'язності еліпсоїдом

Таким чином, для знаходження областей зв'язності, що відповідають маркерам індексу, встановимо наступні геометричні критерії:

- відношення ширини маркеру до його висоти (k_{wh});

$$k_{wh} = \text{'MajorAxisLength'} / \text{'MinorAxisLength'};$$

- відношення площі обмежувального прямокутника маркеру до його висоти (k_{smh})

$$k_{smh} = \text{'FilledArea'} / \text{'MinorAxisLength'};$$

- відношення площі обмежувального прямокутника до площі випуклої оболонки, що апроксимує область зв'язності (k_{smsr})

$$k_{smsr} = \text{'FilledArea'} / \text{'ConvexArea'}.$$

Експериментально встановлено, що для безпомилкового визначення маркерів індексу значення критеріїв лежать у таких діапазонах: $k_{wh} \in (2,55; 3,52)$, $k_{smh} \in (18; 34,47)$ та $k_{smsr} \in (0,86; +\infty)$.

Під час виконання роботи необхідно запропонувати власні діапазони або критерії для пошуку маркерів.

Функція `markerfilter`, яка знаходиться в директорії `Model`, виконує низку операцій, що були задекларовані раніше, а саме: виділення позицій маркерів індексу, а також пошук крайніх маркерів (початкового та кінцевого), визначення кута нахилу зображення, впорядкування (сортування) статистичної інформації. На вхід функція приймає бінарне зображення `imgbwin`, виходом є матриця **STATS**, розмірністю 7×7 , де кількість рядків

відповідає характеристикам, а кількість стовпчиків – загальній кількості маркерів. Детальніше: перший та другий рядок – координати центра мас області зв'язності, третій – кут між віссю абсцис та великою віссю еліпсоїда, четвертий – довжина (в пікселях) великої вісі еліпсоїда, п'ятий – довжина (в пікселях) малої вісі еліпсоїда, шостий та сьомий – координати центра мас маркерів індексу с урахуванням подальшого повороту зображення.

Щоб скористатися цією функцією в нашій моделі задіємо блок «MATLAB Fnc». В полі «MATLAB function» введемо назву m-файлу «markerfilter», а в полі «Output dimensions» проставимо розмір [7, 7]. Проконтролюємо, щоб була знята галочка «Collapse 2-D results to 1-D». Після всіх маніпуляції блок «Detect Markers» має виглядати так як показано на рис. 13.

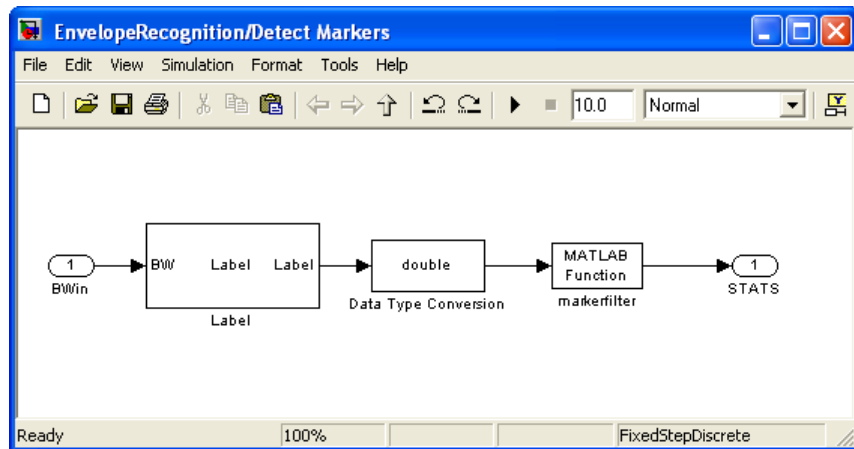


Рисунок 13 – Зовнішній вигляд завершеного блоку «Detect Markers»

Відмітимо, що задача виділення позицій маркерів індексу розв'язується функцією `extractmarkers`, що входить до складу m-файлу `markerfilter`.

Знаходження початкового та кінцевого маркеру

Результатом попереднього етапу є множина регіонів, які відповідають маркерам індексу. Наступними операціями для підготовки до безпосереднього розпізнавання індексу на конверті мають бути поворот зображення на деякий кут (для того, щоб цифри індексу розташовувалися горизонтально) та сегментація цифр індексу. Для повороту зображення треба визначити кут нахилу, а для цього потрібно знайти початковий та кінцевий маркери. Початковим будемо вважати такий маркер, який має біля себе в якості сусіда додатковий маркер (рис. 14).

Може виникнути питання, для чого нам визначати кут нахилу по початковому та кінцевому маркерам, якщо нам відомий кут між віссю абсцис та великою віссю еліпсоїда. Треба відзначити, що цей кут обмежений значеннями в діапазоні від 0 до 180°, а індекс може бути повернутий на кут від 0 до 360°.

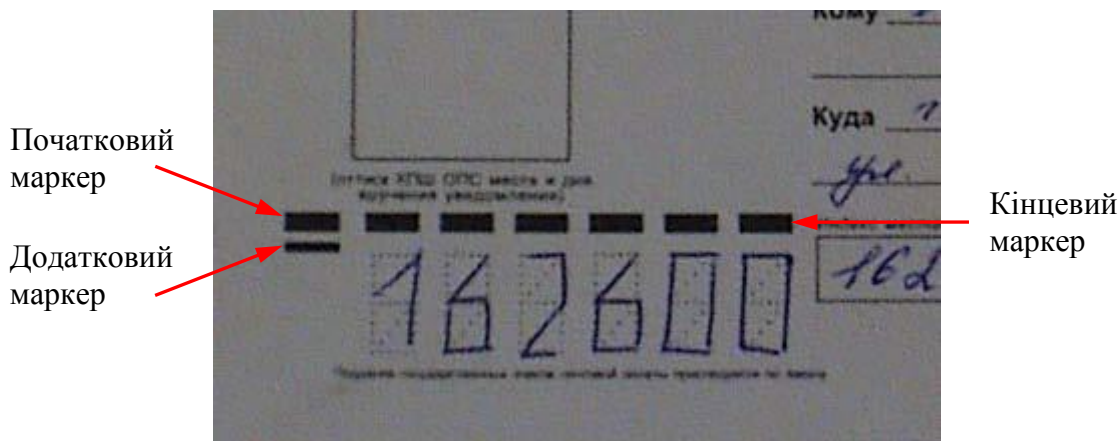


Рисунок 14 – Ключові маркери на зображенні індексу

Спільною ознакою початкового та кінцевого маркерів є те, що вони мають по одному горизонтальному маркеру-сусіду, виняткова ознака початкового маркеру полягає в тому, що біля нього повинен знаходитися додатковий маркер (орієнтація в вертикальній площині) див. рис. 14.

Експериментально встановлено, що для безпомилкової фільтрації додаткового маркеру індексу значення критеріїв лежать у таких діапазонах: $k_{wh} \in [4,54; 7,2]$, $k_{smh} \in [17; 34]$ та $k_{smr} \in [0,8; +\infty)$.

Задача знаходження початкового та кінцевого маркеру реалізується функцією `findendings`, що є складовою m-файла `markerfilter`.

Визначення кута нахилу

Результатом виділення позицій маркерів індексу є статистична інформація, яка включає в себе зокрема дані про орієнтацію маркерів фільтру (характеристика '**Orientation**'). Скористаємося усередненою характеристикою шляхом сумування всіх кутів нахилу і подальшим діленням на загальну кількість маркерів індексу, а саме θ_c :

$$\theta_c = \frac{\sum_{i=1}^K \theta_i}{K},$$

де θ_i – кут нахилу i -го маркеру, K – загальна кількість маркерів ($K = 7$).

Координати початкового і кінцевого маркерів дозволяють з урахуванням усередненої характеристики θ_c визначити повний кут нахилу зображення (функція `definetiltangle`).

Впорядкування розташування маркерів

Характеристики областей на зображенні, де знаходяться маркери індексу, не є впорядкованими, тобто порядкові індекси маркерів не

співпадають з інформацією в матриці **stats** (наприклад, інформація, що знаходиться у другому стовпчику може відноситися до третього маркеру тощо). Впорядкування матриці **stats** може проходити в два етапи:

- визначаємо нові координати центроїд маркерів індексу після повороту;
- сортуємо стовпчики матриці **stats** в порядку збільшення x -координати центроїд маркерів індексу.

Перший етап передбачає використання геометричних просторових перетворень. Нехай зображення f , задане в координатній системі (w, z) , піддається геометричній деформації, в результаті якої виходить зображення g в координатній системі (x, y) . Це (координатне) перетворення можна подати у вигляді формули:

$$(x, y) = T[(w, z)].$$

Наприклад, якщо $(x, y) = T[(w, z)] = (w / 2, z / 2)$, то «спотворення» зводиться до подвійного стискування (зменшення масштабу) в обох просторових вимірах – по ширині та висоті.

Найчастіше використовується клас геометричних перетворень, представники якого називаються афінними перетвореннями. Афінне перетворення можна записати в матричній формі:

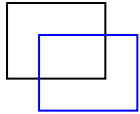
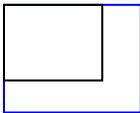
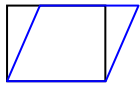
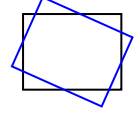
$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} w & z & 1 \end{bmatrix} \mathbf{T} = \begin{bmatrix} w & z & 1 \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

Даною формулою можна задати стискування, поворот, перенесення або зсув, відповідним чином визначаючи елементи матриці **T** (див. табл. 1).

Отже, координатне рівняння операції повороту однозначно дозволяє визначити нові координати центроїд маркерів індексу, після чого їх відбувається впорядкування по зростанню x -координат (функція `arrangestats`, `m`-файл `markerfilter`).

Таким чином, на даному етапі інформація про місцезнаходження маркерів індексу вже є відомою. Для того, щоб перевірити її адекватність виведемо координати центрів мас маркерів на вихідне кольорове зображення. Таким чином, змінимо модель так, як показано на рис. 15. Для цього додамо блок «Submatrix» для виділення перших двох рядків з матриці «**stats**» та блок «Draw Markers» для виводу графічних позначок на зображення. Блоки «Data Type Conversion» та «Stop Simulation» використовуються для перетворення типу `double` в `int32` та закінчення сеансу моделювання відповідно.

Таблиця 1 – Види афінних перетворень

Тип	Афінна матриця	Координатне рівняння	Приклад
Перенесення	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$	$\begin{aligned} x &= w + t_{31} \\ y &= z + t_{32} \end{aligned}$	
Розтягнення	$\begin{bmatrix} t_{11} & 0 & 0 \\ 0 & t_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x &= t_{11} \cdot w \\ y &= t_{22} \cdot z \end{aligned}$	
Зсув	$\begin{bmatrix} 1 & t_{12} & 0 \\ t_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x &= w + t_{21} \cdot z \\ y &= t_{12} \cdot w + z \end{aligned}$	
Поворот	$\begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x &= w \cos \theta - z \sin \theta \\ y &= w \sin \theta + z \cos \theta \end{aligned}$	

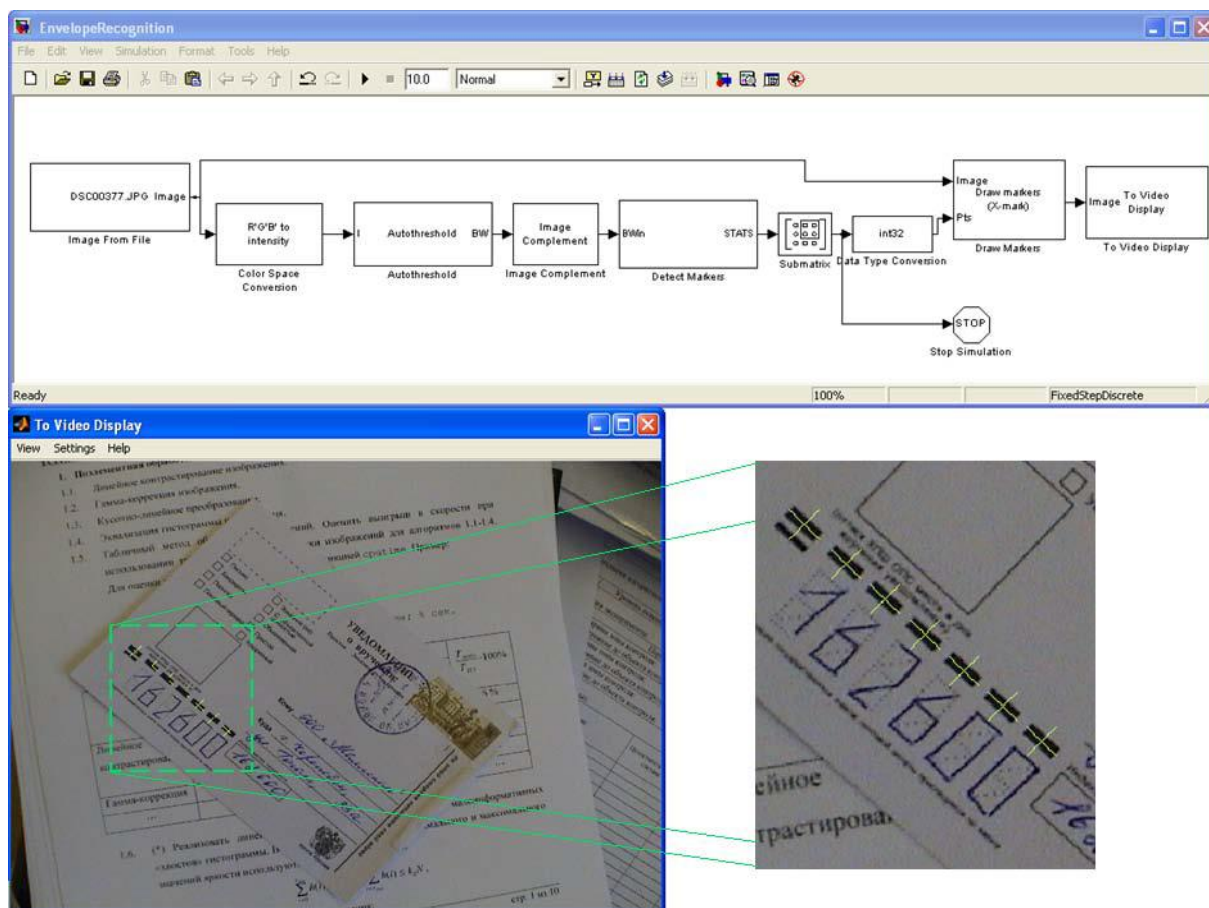


Рисунок 15 – Перевірка точності пошуку центрів маркерів індексу

Поворот зображення

Як ми вже встигли впевнитися, блок «Detect Markers» визначає координати маркерів індексу. Для розпізнавання безпосередньо індексу на конвертах треба провести сегментацію зображення індексу, тобто виділити зображення цифр. З цією метою додамо в нашу модель блок з назвою «Segmentation» (тип «Subsystem»), який буде відповідати за операцію сегментації (рис. 16). Він має включити в себе розв'язання таких задач: поворот зображення, виділення зони індексу та сегментування цієї зони. Блок має такі входи: «**STATS**» – статистична інформація про маркери, «**Image**» – зображення в шкалі сірого. На виході розташувалися такі порти: «**Pts**» – координати центрів маркерів, «**IndexImage**» – зображення зони індексу, «**Rects**» – прямокутники, що описують області цифр, «**Stop**» – прапорець закінчення моделювання та «**Thresh**» – адаптивний поріг для подальшої бінарізації. Зверніть увагу на те, що на рис. 16 відображена модифікація моделі: з'явився блок «Image Preprocess», який об'єднує блоки «Color Space Conversion», «Autothreshold» та «Image Complement».

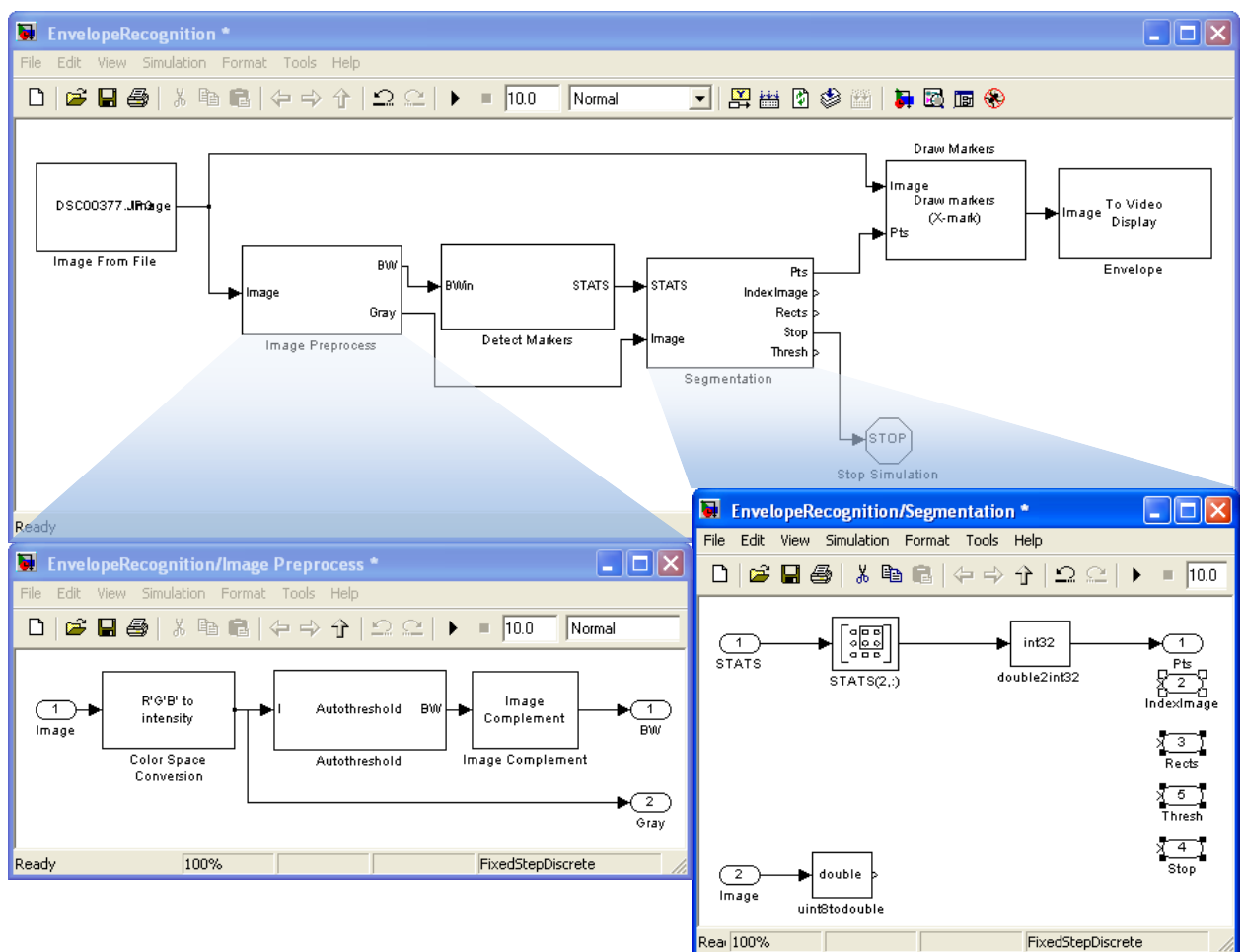


Рисунок 16 – Блок сегментації

Поворот зображення як множини пікселів виконується відповідно до розглянутих вище афінних перетворень. При цьому яскравість кожної точки зображення визначається шляхом інтерполяції.

Інтерполяція – в обчислювальній математиці спосіб знаходження проміжних значень величини по наявному дискретному набору відомих значень.

При виконанні повороту зображень використовуватимемо бікубічну інтерполяцію. Бікубічна інтерполяція – це розширення кубічної інтерполяції на випадок функції двох змінних, значення якої задані на двовимірній регулярній сітці. Поверхня, отримана в результаті бікубічної інтерполяції є гладкою функцією, на відміну від поверхонь, отриманих в результаті білінійної інтерполяції або інтерполяції методом найближчого сусіда. Бікубічна інтерполяція використовується в обробці зображень, даючи якісніше зображення в порівнянні з білінійною фільтрацією.

Припустимо, що необхідно інтерпольовати значення функції $f(x,y)$ в точці $p(x,y)$, що лежить усередині квадрата $[0, 1] \times [0, 1]$, і відоме значення функції f у шістнадцяти сусідніх точках (x, y) , $x = -1, \dots, 2$, $y = -1, \dots, 2$. Тоді загальний вид функції, що задає інтерпольовану поверхню, може бути записаний таким чином:

$$p(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j .$$

Для знаходження коефіцієнтів a_{ij} необхідно підставити в наведене вище рівняння значення функції у відомих шістнадцяти точках. Наприклад:

$$\left\{ \begin{array}{l} f(-1,-1) = a_{00} - a_{01} + a_{02} - a_{03} - a_{10} + a_{11} - a_{12} + a_{13} + a_{20} - \\ \quad - a_{21} + a_{22} - a_{23} - a_{30} + a_{31} - a_{32} + a_{33} \\ \quad \vdots \\ f(-1,0) = a_{00} - a_{10} + a_{20} - a_{30} \\ f(0,0) = a_{00} \\ f(1,0) = a_{00} + a_{10} + a_{20} + a_{30} \\ f(2,0) = a_{00} + 2a_{10} + 4a_{20} + 8a_{30} \\ \quad \vdots \\ f(1,1) = a_{00} + a_{01} + a_{02} + a_{03} + a_{10} + a_{11} + a_{12} + a_{13} + a_{20} + \\ \quad + a_{21} + a_{22} + a_{23} + a_{30} + a_{31} + a_{32} + a_{33} \\ \quad \vdots \\ f(2,2) = a_{00} + 2a_{01} + 4a_{02} + 8a_{03} + 2a_{10} + 4a_{11} + 8a_{12} + 16a_{13} + \\ \quad + 4a_{20} + 8a_{21} + 16a_{22} + 32a_{23} + 8a_{30} + 16a_{31} + 32a_{32} + 64a_{33} \end{array} \right.$$

Повністю в матричному вигляді:

$$\mathbf{M}\alpha^T = \gamma^T$$

де

$$\alpha = (a_{00} \ a_{01} \ a_{02} \ a_{03} \ a_{10} \ a_{11} \ a_{12} \ a_{13} \ a_{20} \ a_{21} \ a_{22} \ a_{23} \ a_{30} \ a_{31} \ a_{32} \ a_{33});$$

$$\gamma = (f(-1,-1) \ f(0,-1) \ f(1,-1) \ f(2,-1) \ f(-1,0) \ f(0,0) \ f(1,0) \ f(2,0) \dots \\ f(-1,1) \ f(0,1) \ f(1,1) \ f(2,1) \ f(-1,2) \ f(0,2) \ f(1,2) \ f(2,2))$$

$$\mathbf{M} = \begin{pmatrix} 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 2 & -2 & 2 & -2 & 4 & -4 & 4 & -4 & 8 & -8 & 8 & -8 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 8 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 4 & 4 & 4 & 4 & 8 & 8 & 8 & 8 \\ 1 & 2 & 4 & 8 & -1 & -2 & -4 & -8 & 1 & 2 & 4 & 8 & -1 & -2 & -4 & -8 \\ 1 & 2 & 4 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 4 & 8 & 1 & 2 & 4 & -8 & 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 \\ 1 & 2 & 4 & 8 & 2 & 4 & 8 & 16 & 4 & 8 & 16 & 32 & 8 & 16 & 32 & 64 \end{pmatrix}$$

Розв'язуючи отриману систему лінійних алгебраїчних рівнянь, можна знайти a_{ij} в явному вигляді:

$$\alpha^T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 36 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -12 & 0 & 0 & 0 & -18 & 0 & 0 & 0 & 36 & 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & -36 & 0 & 0 & 0 & 18 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 & 0 & 18 & 0 & 0 & 0 & -18 & 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & -12 & -18 & 36 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & -12 & 2 & 6 & 9 & -18 & 3 & -12 & -18 & 36 & -6 & 2 & 3 & -6 & 1 \\ -6 & -9 & 18 & -3 & 12 & 18 & -36 & 6 & -6 & -9 & 18 & -3 & 0 & 0 & 0 & 0 \\ 2 & 3 & -6 & 1 & -6 & -9 & 18 & -3 & 6 & 9 & -18 & 3 & -2 & -3 & 6 & 1 \\ 0 & 0 & 0 & 0 & 18 & -36 & 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -6 & 12 & -6 & 0 & -9 & 18 & -9 & 0 & 18 & -36 & 18 & 0 & -3 & 6 & -3 & 0 \\ 9 & -18 & 9 & 0 & -18 & 36 & -18 & 0 & 9 & -18 & 9 & 0 & 0 & 0 & 0 & 0 \\ -3 & 6 & -3 & 0 & 9 & -18 & 9 & 0 & -9 & 18 & -9 & 0 & 3 & -6 & 3 & 0 \\ 0 & 0 & 0 & 0 & -6 & 18 & -18 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & -6 & 6 & -2 & 3 & -9 & 9 & -3 & -6 & 18 & -18 & 6 & 1 & -3 & 3 & -1 \\ -3 & 9 & -9 & 3 & 6 & -18 & 18 & -6 & -3 & 9 & -9 & 3 & 0 & 0 & 0 & 0 \\ 1 & -3 & 3 & -1 & -3 & 9 & -9 & 3 & 3 & -9 & 9 & -3 & -1 & 3 & -3 & 1 \end{pmatrix} x^T$$

Одного разу знайдені коефіцієнти a_{ij} тепер можуть бути використані для багатократного обчислення інтерпольованого значення функції в довільних точках квадрата $[0, 1] \times [0, 1]$.

Процедуру, яка повертає зображення на визначений кут, виділяє зони індексу та сегментів з цифрами, легше всього реалізувати у вигляді m-файлу, який розміщується в спеціальному блоці типу «MATLAB Fcn». Однак при такому підході виникає проблема, яка полягає у тому, що m-файл зможе приймати та повертати лише по одному елементу. Водночас, для задачі необхідно два вхідних аргументи («**STATS**» – статистична інформація про маркери, «**Image**» – зображення в шкалі сірого) та три вихідних («**IndexImage**» – зображення зони індексу, «**Rects**» – прямокутники, що описують області цифр, «**Thresh**» – адаптивний поріг для подальшої бінарізації). Отже, застосуємо невелику хитрість – переведемо вхідні аргументи (двовимірні масиви «**STATS**» та «**Image**») у вектори та додамо в кінці цих векторів розміри двовимірних масивів. Такий перевід реалізуємо за допомогою m-файлу «transform2Dto1D», який упаковується в блок «MATLAB Fcn». Потім зробимо конкатенацію цих векторів, а результат будемо подавати на вхід блоку «MATLAB Fcn», який за допомогою m-файлу segmentindexes виконує вищеописану процедуру (рис. 17).

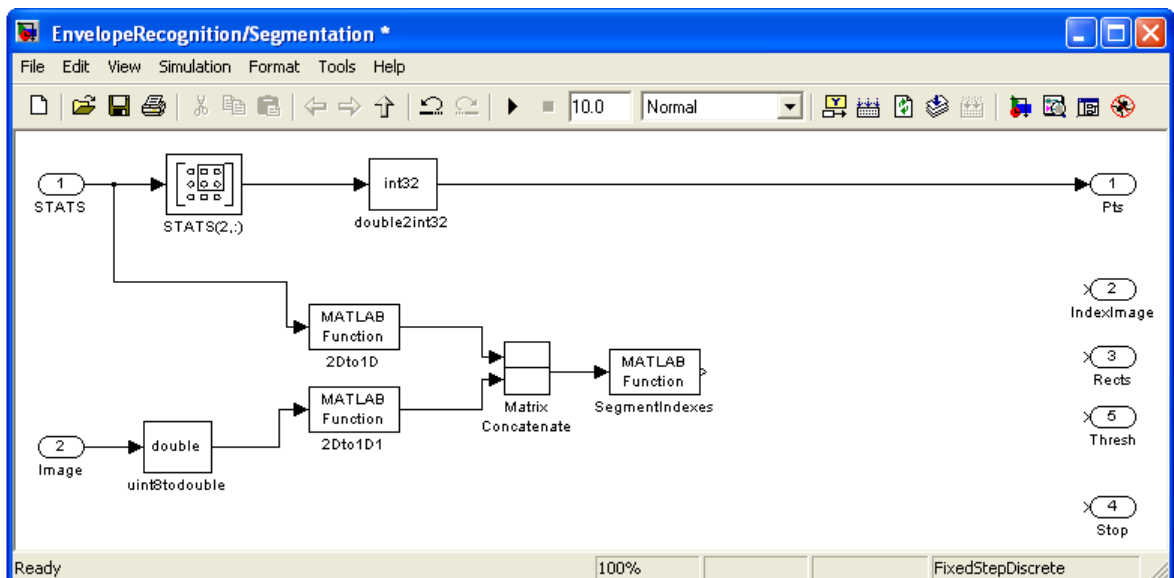


Рисунок 17 – Блок сегментації

Виділення зони індексу

Процедуру виділення зони індексу ілюструє рис. 18. Нам відома інформація про ширину та висоту (w, h) маркерів та координати їх центроїд. На основі цієї інформації легко визначити координати прямокутника, який окреслює зону індексу. **Увага!** Simulink може не працюватиме з блоками, розмірність сигналів на виході яких є змінною величиною. Отже, після виділення зображення індексу треба його привести до розмірів [65, 217]. **Будьте уважні!** Зміна розмірів повинна враховуватися при сегментуванні зони індексу.

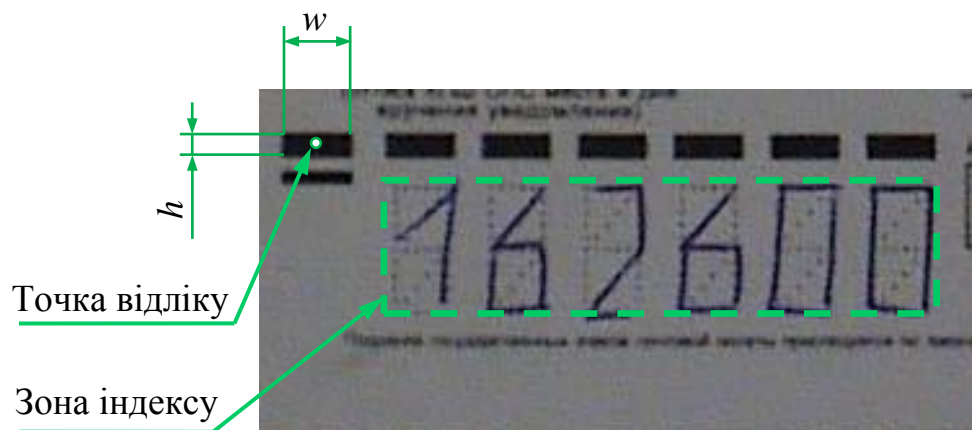


Рисунок 18 – Виділення зони індексу

Розв'язання задачі виділення зони індексу відбувається за допомогою m-файлу segmentindexes.

Сегментування зони індексу

Задачею сегментації є знаходження прямокутників, що описують області цифр індексу (рис. 19). Маючи координати кожного маркера та їх розміри, можна визначити координати прямокутників. Розв'язання цієї задачі також виконується m-файлом `segmentindexes`.

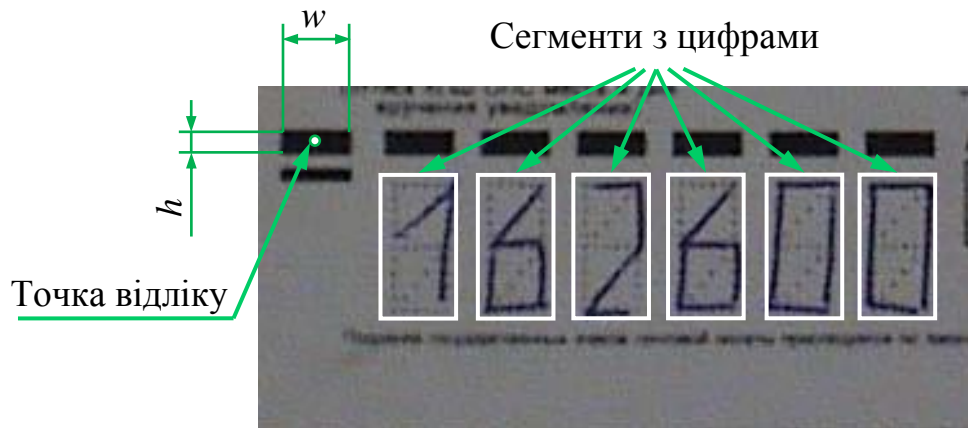


Рисунок 19 – Сегментування зони індексу

Як було сказано раніше, на виході блоку «`SegmentIndexes`» маємо одномірний сигнал розміром $[65 \times 217 + 4 \times 7 + 1, 1]$, який включає зображення зони індексу (`IndexImage`), прямокутники, що описують області цифр (`Rects`) та адаптивний поріг для подальшої бінарізації (`Thresh`). «Розкодування» такого сигналу виконаємо за допомогою трьох блоків «`MATLAB Fnc`», які містять такі m-файли: `extractindeximage`, `extractrects` та `extractthresh` (див. рис. 20).

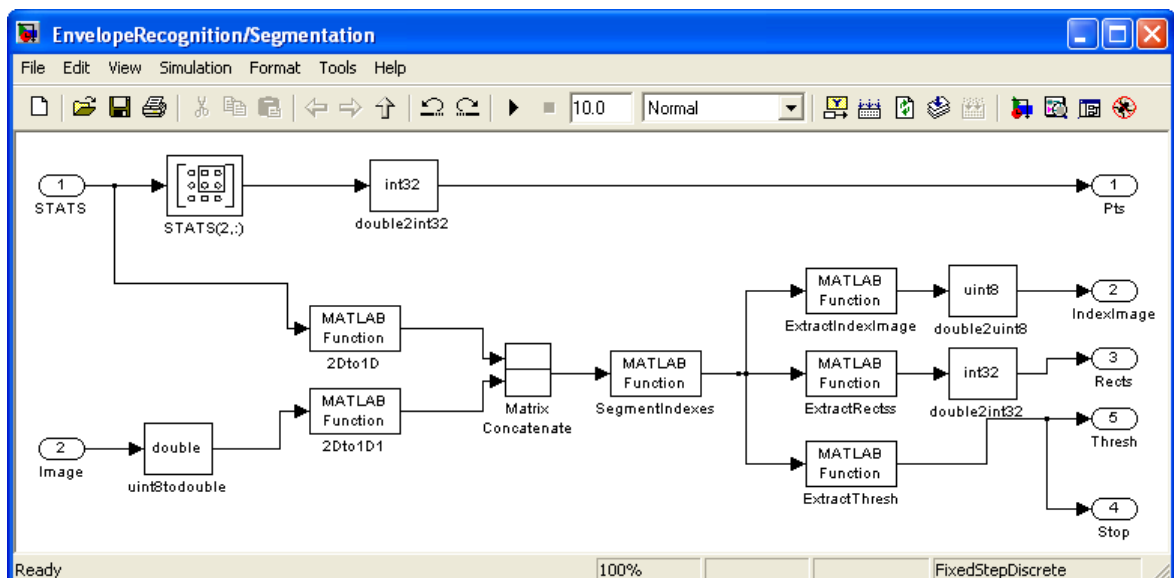


Рисунок 20 – Остаточний вигляд блоку сегментації цифр індексу

Для перевірки коректності його роботи додамо в нашу модель блок прорисовки прямокутників «`Draw Shapes`» та блок відображення зображення

індексу (рис. 21). Після запуску моделювання отримуємо одне зображення з поміченими індексами, а друге – із зоною індексу та сегментованими цифрами.

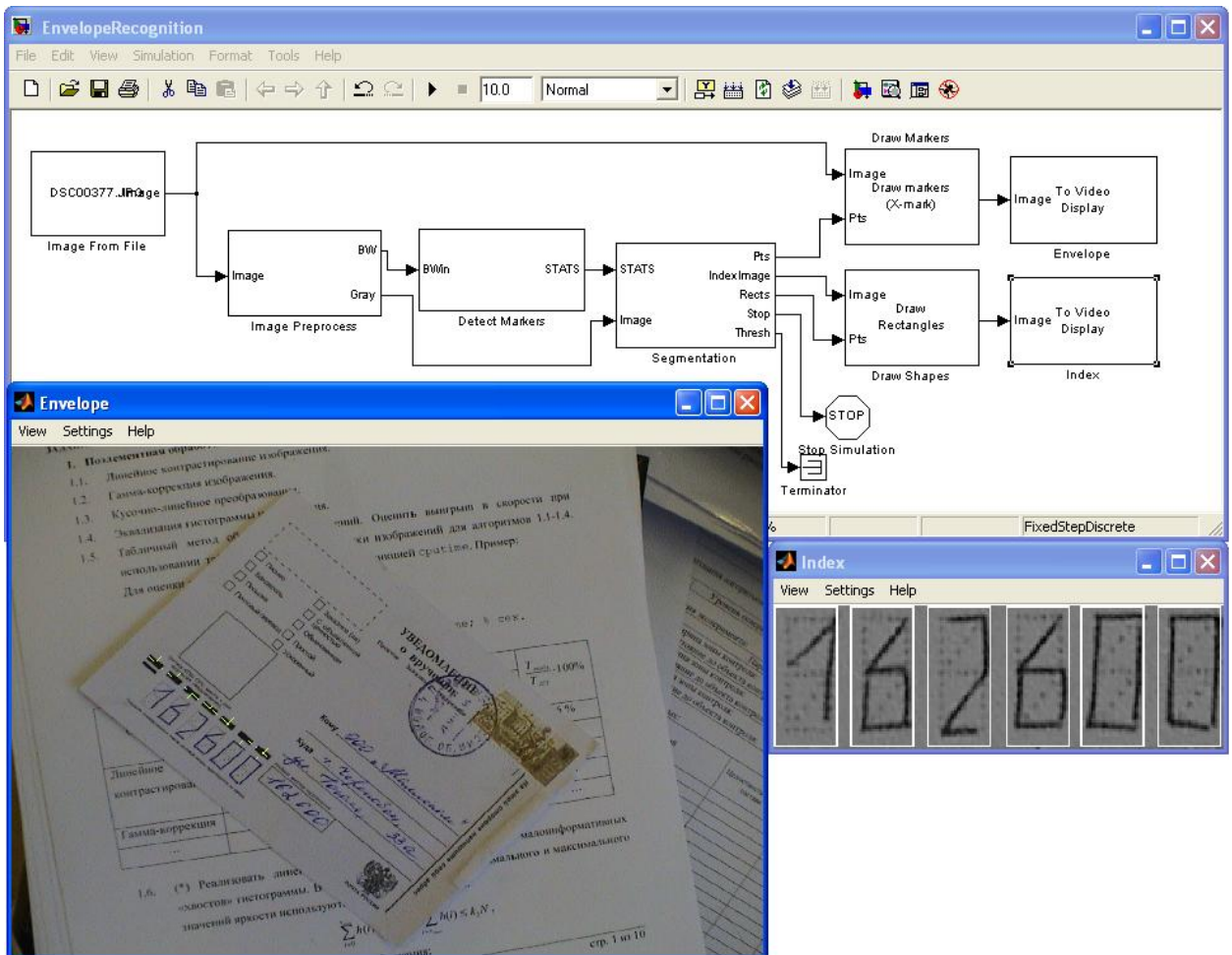


Рисунок 21 – Результат сегментації цифр індексу

Розпізнавання зони індексу

Для розв’язання останньої задачі скористуємося неймережними технологіями, які реалізовані в системі МАТЛАВ (Simulink) [4]. Нейронні мережі володіють як перевагами, так і недоліками, проте вони представляють зручний інструмент для вирішення задач розпізнавання образів.

Простий нейрон

Елементарною коміркою нейронної мережі є *нейрон*. Структура нейрона з єдиним скалярним входом показана на рис. 22, а.

Скалярний вхідний сигнал p помножується на скалярний ваговий коефіцієнт w і результуючий зважений вхід $n = w \cdot p$ передається як аргумент функції активації нейрона f , яка породжує скалярний вихід a .

Нейрон, показаний на рис. 22, б, доповнений скалярним зміщенням b . Зміщення підсумовується із зваженим входом $w \cdot p$ і приводить до зсуву аргументу функції f на величину b .

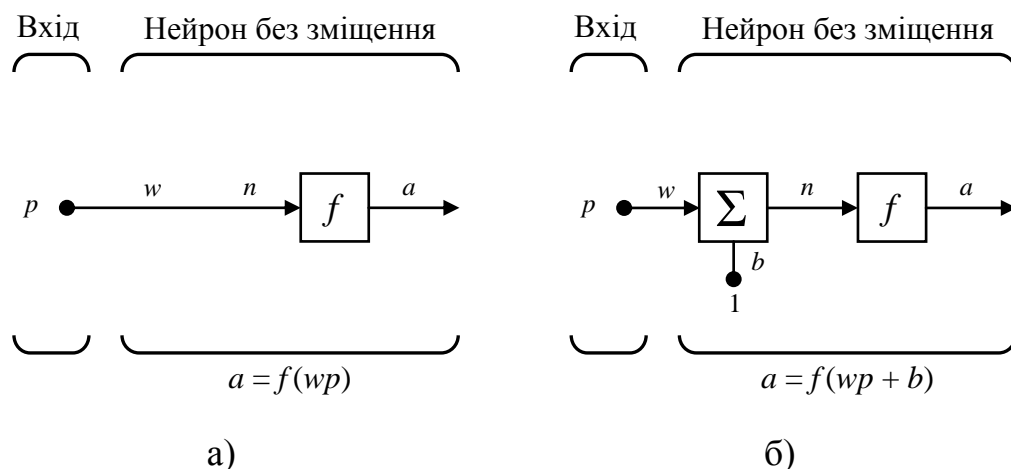


Рисунок 22 – Приклад моделі штучного нейрона:

a – без зміщення; b – зі зміщенням

Дію зсуву можна звести до схеми зважування, якщо уявити, що нейрон має другий вхідний сигнал із значенням, рівним 1. Вхід n функції активації нейрона як і раніше залишається скалярним і рівним сумі зваженого входу і зміщення b . Ця сума є аргументом функції активації f ; виходом функції активації є сигнал a . Константи w і b є скалярними параметрами нейрона. Основний принцип роботи нейронної мережі полягає в налаштуванні параметрів нейрона так, щоб відгук мережі на певні вхідні сигнали відповідав деякій бажаній поведінці. Регулюючи ваги або параметри зміщення, можна навчити мережу виконувати конкретну роботу; також можливо, щоб мережа сама коректувала свої параметри для досягнення необхідного результату самостійно.

Функції активації

Функції активації (передавальні функції) нейрона можуть мати самий різноманітний вигляд. Функція активації f , як правило, належить до класу сигмоїдних функцій з аргументом n і виходом a .

Розглянемо три найбільш поширені форми функції активації:

- *Одинична функція активації з жорстким обмеженням $\text{hardlim}(n)$.* Ця функція описується співвідношенням $a = \text{hardlim}(n) = 1(n)$ (функція Хевісайда) і показана на рис. 23, а. Вона дорівнює 0, якщо $n < 0$ та 1, якщо $n \geq 0$.
- *Лінійна функція активації $\text{purelin}(n)$.* Дана функція описується співвідношенням $a = \text{purelin}(n) = n$ і показана на рис. 23, б.

- *Логістична функція активації* $\text{logsig}(n)$. Ця функція описується співвідношенням $a = \text{logsig}(n) = 1 / (1 + \exp(-n))$ і показана на рис. 23, в. Вона належить до класу сигмоїдальних функцій, і її аргумент може набувати будь-якого значення в діапазоні від $-\infty$ до $+\infty$, а вихід змінюється в діапазоні від 0 до +1.

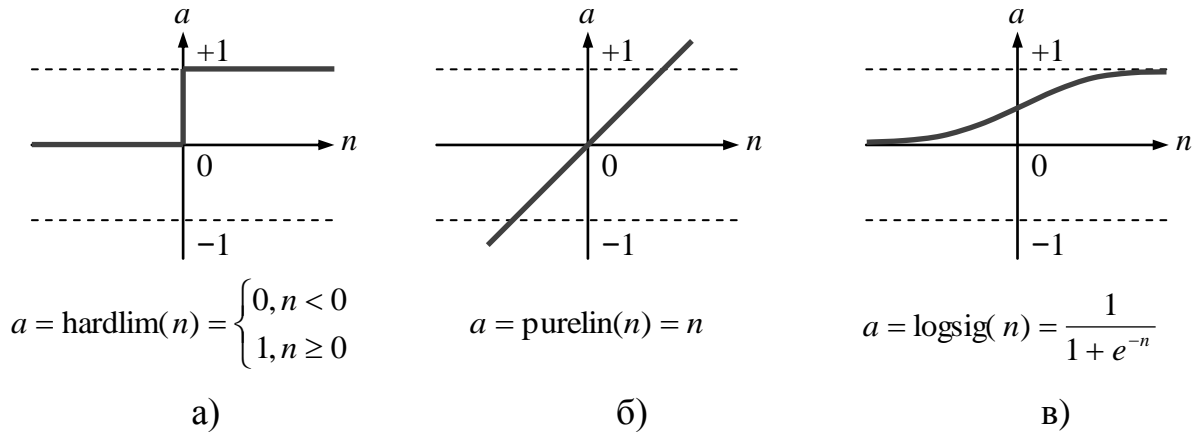


Рисунок 23 – Функції активації нейронів:

- а – одинична функція активації з жорстким обмеженням;
- б – лінійна функція активації;
- в – логістична функція активації

В МАТЛАВ включені і інші функції активації. Використовуючи мову МАТЛАВ, користувач може створювати і свої власні унікальні функції.

Нейрон з векторним входом

Нейрон з одним вектором входу \mathbf{p} , що містить R елементів p_1, p_2, \dots, p_R показано на рис. 24. Тут кожен елемент входу помножується на ваги w_1, w_2, \dots, w_R відповідно і зважені значення передаються на суматор. Їх сума дорівнює скалярному добутку вектора-рядка \mathbf{W} на вектор входу \mathbf{p} .

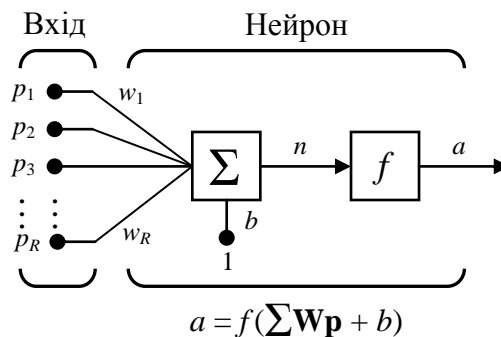


Рисунок 24 – Нейрон з векторним входом

Нейрон має зміщення b , яке підсумовується із зваженою сумою входів. Результуюча сума n рівна $n = w_1 \cdot p_1 + w_2 \cdot p_2 + \dots + w_R \cdot p_R + b$ і служить

аргументом функції активації f . У нотації мови MATLAB цей вираз записується наступним чином: $\mathbf{n} = \mathbf{W} * \mathbf{p} + \mathbf{b}$.

Структура нейрона, показана вище, занадто деталізована, тому при розгляді мереж, що складаються з великого числа нейронів, зручніше користуватися укрупненою структурною схемою нейрона (див. рис. 25).

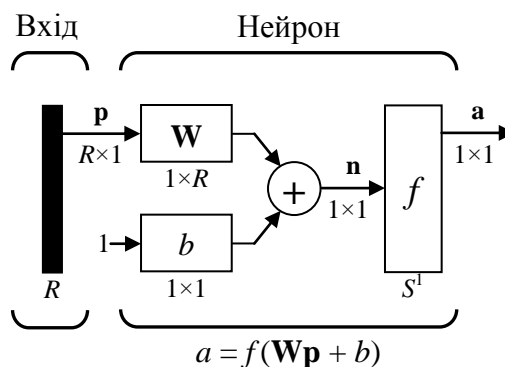


Рисунок 25 – Укрупнена структурна схема нейрона з векторним входом

Вхід нейрона замальовується у вигляді темної вертикальної межі, під якою вказується кількість елементів входу R . Розмір вектора входу \mathbf{p} вказується нижче символу \mathbf{p} , він рівний $R \times 1$. Вектор входу множиться на вектор-рядок \mathbf{W} довжини R . Як і раніше, константа 1 розглядається як вхід, що помножується на скалярне зміщення b . Входом n функції активації нейрона служить сума зміщення b і добутку $\mathbf{W} \cdot \mathbf{p}$. Ця сума подається на вхід функції активації f , на виході якої формується вихід нейрона a , що є скалярною величиною. Структурна схема, наведена на рис. 25, називається шаром мережі. Шар характеризується матрицею ваг \mathbf{W} , зміщенням b , операціями множення $\mathbf{W} \cdot \mathbf{p}$, підсумовування і функцією активації f . Вектор входів \mathbf{p} зазвичай не включається в характеристики шару.

Кожного разу, коли використовується скорочене позначення мережі, розмірність матриць вказується під іменами векторно-матричних змінних. Ця система позначень ілюструє будову мережі і пов'язану з нею матричну математику.

Архітектура нейронних мереж

Реальна нейронна мережа може містити один або більшу кількість шарів і відповідно характеризуватися як одношарова або як багатошарова.

Одношарова нейронна мережа

Розгорнута схема мережі з одним шаром, що має R вхідних елементів та S нейронів показана на рис. 26.

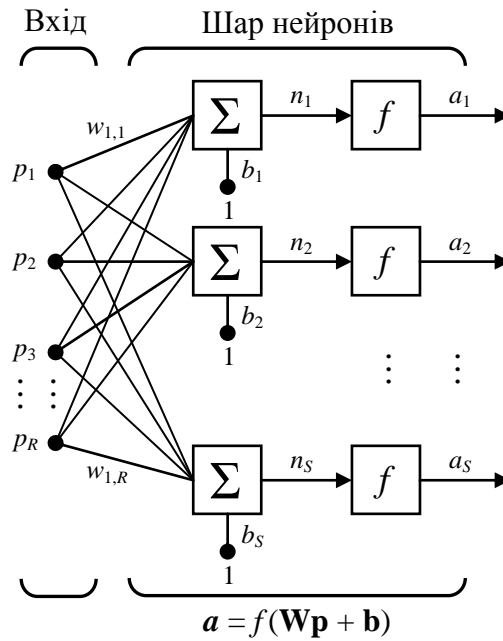


Рисунок 26 – Одношарова нейронна мережа

В наведені вище мережі кожен елемент вектора входу сполучений з усіма входами нейрона і це з'єднання задається матрицею ваг \mathbf{W} ; при цьому кожен i -й нейрон включає суматор, який формує скалярний вихід $n(i)$. Сукупність скалярних функцій $n(i)$ об'єднується в S -елементний вектор входу \mathbf{n} функції активації шару. Виходи шару нейронів формують вектор-стовпець \mathbf{a} , і, таким чином, опис шару нейронів має вигляд:

$$\mathbf{a} = f(\mathbf{W}\mathbf{p} + \mathbf{b})$$

Кількість входів R у шарі може не збігатися з кількістю нейронів S . У кожному шарі, як правило, використовується одна і та ж функція активації. Проте можна створювати складені шари нейронів, в яких використовуються різні функції активації. При цьому нейрони сполучатимуться паралельно. Така мережа матиме однакову входів для кожного нейрона і кожен з них генеруватиме певну кількість виходів. Елементи вектора входу передаються в мережу через матрицю ваг \mathbf{W} , що має вигляд:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \dots & \dots & \dots & \dots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix}$$

Відмітимо, що індекси рядків матриці \mathbf{W} вказують на нейрон, якому належить конкретна вага, а індекси стовпців – яке джерело є входом для даної ваги. Таким чином, елемент матриці ваг $w_{1,2} = \mathbf{W}(1, 2)$ визначає

коефіцієнт, на який множиться другий елемент входу при передачі його на перший нейрон мережі.

Для одношарової мережі з S нейронами укрупнена структурна схема показана на рис. 27.

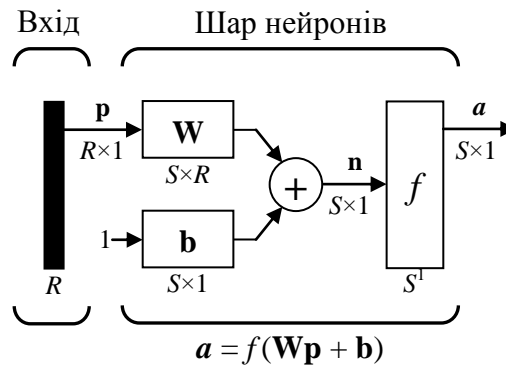


Рисунок 27 – Одношарова нейронна мережа (укрупнена модель представлення)

Тут p – вектор входу розміру $R \times 1$; W – вагова матриця розміру $S \times R$; a , b , n – вектори розміру $S \times 1$.

Багатшарові нейронні мережі

Розглянемо мережі, що мають декілька шарів. Називатимемо вагові матриці, сполучені з входами, *вагами входу шару*, а вагові матриці для сигналів, які витікають з шару, називатимемо *вагами виходу шару*. Також для задання джерела та адресату ваг, зміщень та функцій активації використовуватимемо верхні індекси. Для ілюстрації розглянемо спочатку тільки один, перший шар багатшарової мережі (рис. 28).

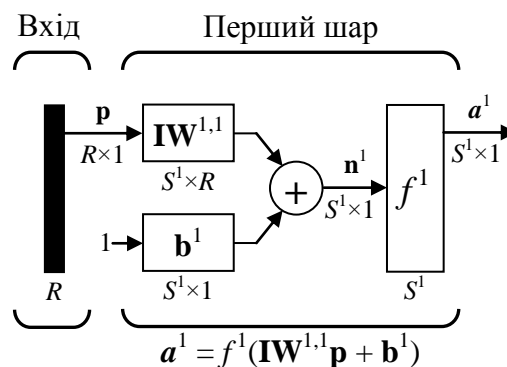


Рисунок 28 – Перший шар багатшарової мережі

Позначимо вагову матрицю, пов'язану з входами, через $IW^{1,1}$, верхні індекси якої указують, що джерелом входів є перший шар (другий індекс), і адресатом також є перший шар (перший індекс). Елементи цього шару є наступними: зміщення b^1 , вхід функції активації n^1 і вихід шару a^1 , мають верхній індекс 1, щоб позначити, що вони належать до першого шару. Надалі

для матриць ваг входу і виходу шару будуть використані позначення **IW** (Input Weight) і **LW** (Layer Weight) відповідно.

Коли мережа має декілька шарів, то кожен шар має свою матрицю ваг **W**, вектор зміщення **b** і вектор виходу **a**. Щоб розрізняти вагові матриці, вектори виходу та інші елементи мережі окремих шарів, використовуватимемо номер у верхньому індексі кожної змінної. Використання цієї системи позначень для мережі з трьох шарів можна бачити на показаній нижче структурній схемі, а також у рівняннях, наведених під нею рис. 29.

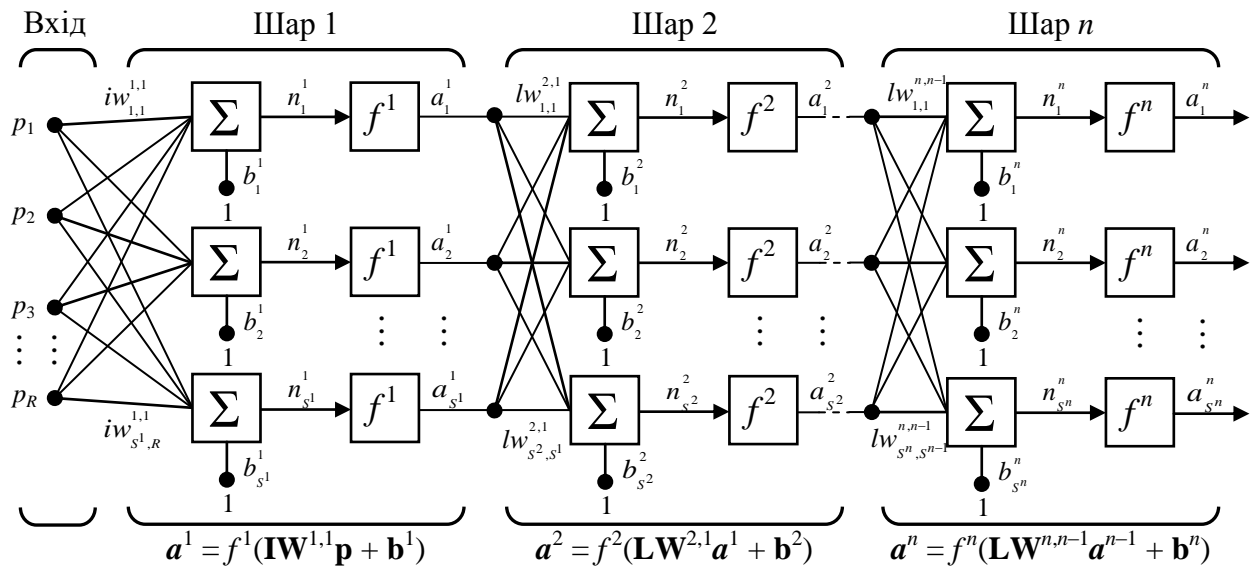


Рисунок 29 – Тришарова нейронна мережа

Мережа, показана вище, має R входів, S^1 нейронів в першому шарі, S^2 нейронів в другому шарі і так далі. Для спільності вважатимемо, що різні шари мають різне число нейронів. На зміщення для кожного нейрона поданий постійний вхідний сигнал 1. Відмітимо, що виходи кожного проміжного шару служать входами для наступного шару. Таким чином, шар 2 може бути розглянутий як один шар мережі з S^1 входами, S^2 нейронами і $S^1 \times S^2$ матрицею ваг **W**². Вхід до шару 2 є 1, а вихід – 2. Тепер, коли позначені всі вектори і матриці шару 2, можна трактувати його як окрему одношарову мережу. Такий підхід може бути застосований до будь-якого шару мережі.

Шари багатошарової мережі мають різні призначення. Шар, який утворює вихід мережі, називається шаром виходу. Всі інші шари називаються прихованими шарами. Тришарова мережа, показана вище, має вихідний шар (шар 3) і 2 прихованих шари (шар 1 і шар 2).

Багатошарові мережі володіють потужними можливостями. Зокрема, двошарова мережа, в якій у першому шарі застосовується сигмоїдальна, а у

другому – лінійна функції активації, може бути навчена апроксимувати з довільною точністю будь-яку функцію зі скінченним числом точок розриву.

Для тришарової нейронної мережі укрупнена структурна схема може бути представлена так, як показано на рис. 30.

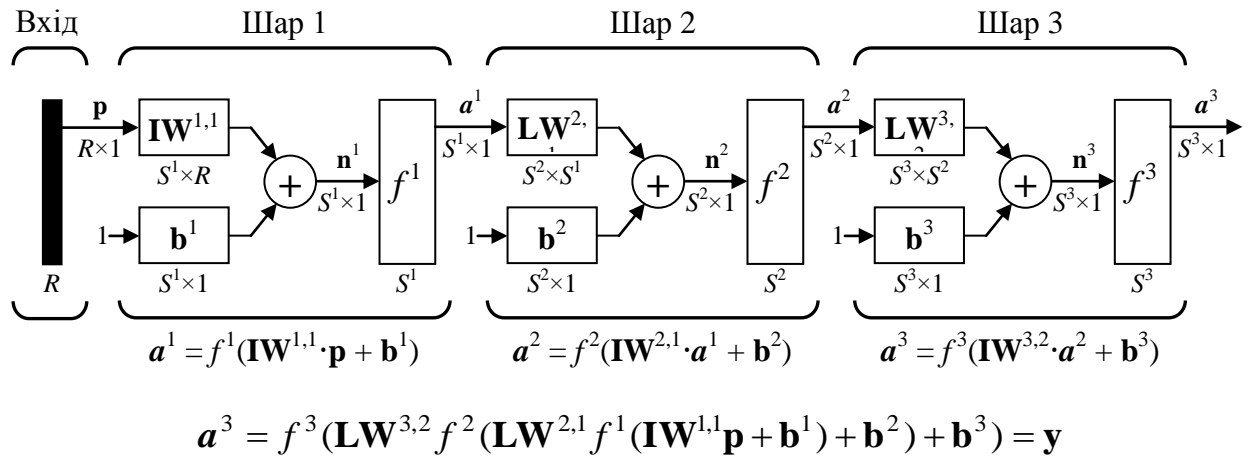


Рисунок 30 – Тришарова нейронна мережа (укрупнена модель представлення)

Формування навчальної вибірки та навчання мережі

В навчальну вибірку входять зображення цифр індексу. Кількість класів повинна дорівнювати 10, тому що використовуються цифри від «0» до «9». В якості прикладів можна взяти ті цифри, що присутні на зображеннях, що розпізнаються (рис. 31).

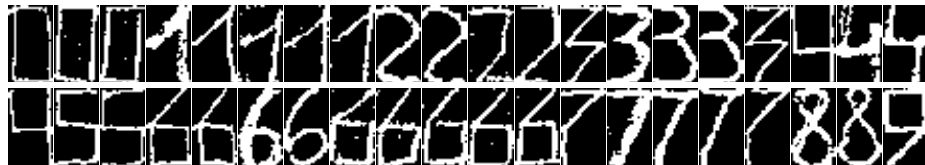


Рисунок 31 – Приклади цифр, що розпізнаються

Отримати подібні зображення можна при наступній послідовності дій:

- 1) провести бінарізацію зображення індексу, використовуючи адаптивне порогове значення;
- 2) використовуючи координати сегментів, вирізати зображення кожної цифри;
- 3) інвертувати зображення, так щоб пікселі фону мали нульове значення.

Нейронна мережа на вхід приймає вектор-стовпець, а зображення цифр є матрицями, отже, треба «розгорнути» матрицю у вектор (наприклад, функцією **reshape**). Якщо зображення цифри має розмір $m \times n$, то результуючий вектор – $(m \cdot n) \times 1$. Для реалізації принципу навчання з вчителем

в системі MATLAB на вхід нейронної мережі подається матриця, яка об'єднує всі приклади цифр (розміром $(m \cdot n) \times N$), а на вихід матриця цілей, яка має розмір $10 \times N$, де 10 – кількість класів, N – кількість прикладів. Отже, якщо необхідно навчити нейронну мережу розпізнаванню таких цифр:



то матриця цілей повинна мати такий вигляд:

$t =$	1	1	1	0	0	0	0	0	← клас «0»
	0	0	0	1	1	1	0	0	← клас «1»
	0	0	0	0	0	0	0	0	← клас «2»
	0	0	0	0	0	0	1	0	← клас «3»
	0	0	0	0	0	0	0	0	← клас «4»
	0	0	0	0	0	0	0	1	← клас «5»
	0	0	0	0	0	0	0	0	← клас «6»
	0	0	0	0	0	0	0	0	← клас «7»
	0	0	0	0	0	0	0	0	← клас «8»
	0	0	0	0	0	0	0	0	← клас «9»

Одиниці в матриці вказують на належність конкретного приклада до класу.

Зауваження! Для успішного навчання та подальшого використання нейронної мережі її слід навчати на досить великій навчальній вибірці (сотні прикладів для кожного символу). Оскільки у нас в наявності є не багато прикладів цифр, їх кількість можна спробувати збільшити, для чого застосувати афінні спотворення та накладання шуму, в такий спосіб штучно розширюючи навчальну вибірку.

Формування архітектури мережі

Перший крок при роботі з нейронними мережами - це створення моделі мережі. Для створення мереж з прямою передачею сигналу в системі MATLAB призначена функція **newff**. Вона має 4 вхідних аргументу і 1 вихідний аргумент – об'єкт класу **network**. Перший вхідний аргумент – це масив розміру $(m \times n) \times 2$, що містить допустимі границі значень (мінімальне і максимальне) для кожного з $(m \times n)$ елементів вектора входу, другий – масив для завдання кількості нейронів кожного шару; третій – масив комірок, що містить імена функцій активації для кожного шару; четвертий – ім'я функції навчання. Наприклад, наступний оператор створює мережу з прямою передачею сигналу:

```
net = newff(minmax(trainset), [100 10], {'tansig', 'tansig'}, 'traincgb');
```

Ця мережа має один прихований шар із 100 нейронами і один вихідний із 10 нейронами; функції активації: *tansig* – у прихованому і вихідному шарах; функція навчання – *traincgb* (метод спряжених градієнтів).

Функція **newff** не лише створює архітектуру мережі, але і ініціалізує її ваги і зміщення, готуючи нейронну мережу до навчання. Проте існують ситуації, коли потрібна спеціальна процедура ініціалізації мережі.

Навчання нейронної мережі

Основний алгоритм зворотного розповсюдження помилки коректує параметри, що настраюються, у напрямі максимального зменшення функціонала помилки. Але такий напрям далеко не завжди є найсприятливішим напрямом, щоб за можливо мале число кроків забезпечити збіжність до мінімуму функціонала. Існують напрями руху, рухаючись по яких можна визначити шуканий мінімум набагато швидше. Зокрема, це можуть бути так звані зв'язані напрями, а відповідний метод оптимізації – це метод спряжених градієнтів.

Для всіх алгоритмів методу зв'язаних градієнтів напрям пошуку періодично встановлюється заново на напрям антиградієнта, або, іншими словами, виконується рестарт. Це відбувається в тих випадках, коли виникають проблеми зі збіжністю. Зокрема, якщо кількість ітерацій перевищено або виникли інші умови, що свідчать про погану збіжність. Одна з таких стратегій рестарту реалізована в алгоритмі CGB, запропонованому Бієле (Beale) і Пауеллом (Powell). Згідно цієї стратегії рестарт виконується, якщо поточний і передуючий напрями градієнтів є неортогональними, і ця умова визначається таким чином:

$$|g_{t-1}^k g_k| \geq 0,2 \cdot \|g_k\|^2.$$

Для навчання мережі в командний рядок введемо наступні оператори:

```
net.trainParam.goal = 1e-3; % цільова помилка
net = train(net, p, t);    % запуск навчання
```

Ознакою того, що мережа навчилася правильно є досягнення цільового значення помилки (рис. 32).

Оскільки навчена нейронна мережа буде далі використовуватись при розпізнаванні, то збережемо її структуру (змінну **net**) у файлі **net.mat**.

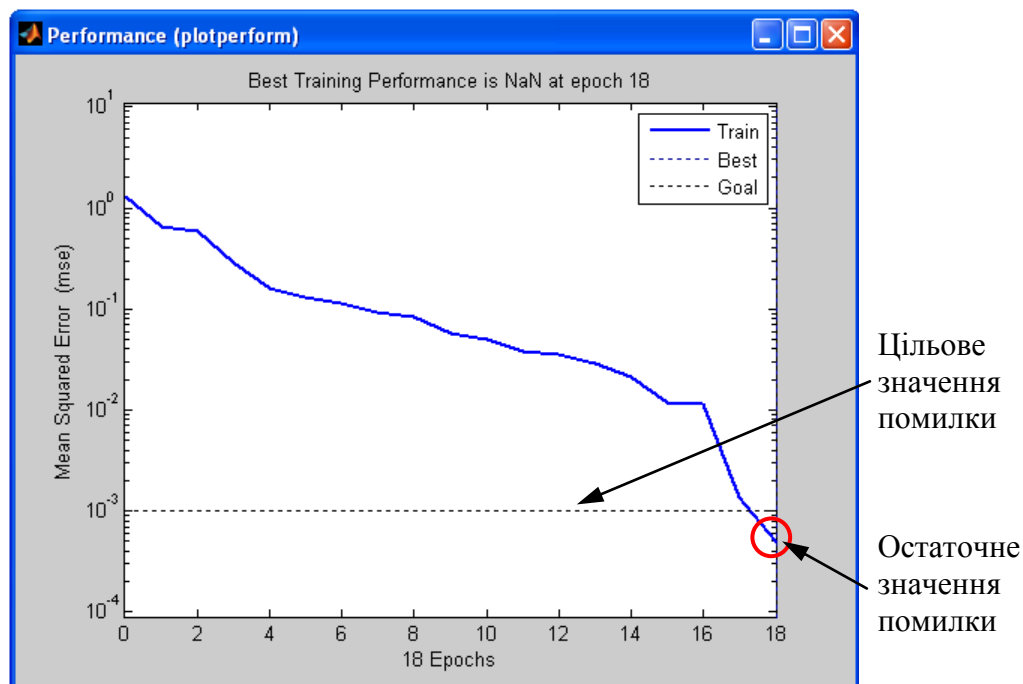


Рисунок 32 – Графік зміни помилки при навчанні нейронної мережі

Розпізнавання цифр індексу

Розпізнавання цифр є останньою задачею. Індекс складається з шести цифр, отже на виході блоку розпізнавання має бути вектор розмірністю [1, 6]. Проектування такого блоку будемо здійснювати таким же самим чином, як і блок з назвою «Segmentation» (тип «Subsystem»). Блок включає такі вхідні порти: «**IndexImage**» – зображення зони індексу, «**Rects**» – прямокутники, що описують області цифр, «**Thresh**» – адаптивний поріг для бінарізації. На виході один єдиний порт «**Index**», який містить значення індексу. Як і в попередньому випадку всі вхідні сигнали запаковані у єдиний одномірний 1D вектор, який передається в блок типу «MATLAB Fcn». М-файл (recognizeindexes.m) цього блоку реалізує наступний алгоритм:

- 1) «розпаковує» вхідний вектор у три змінні;
- 2) завантажує нейронну мережу з файлу net.mat;
- 3) використовує значення **Rects** для виділення із зображення **IndexImage** нових зображень цифр індексу (функція imcrop);
- 4) трансформує кожне зображення цифри у вектор розмірністю $(m \times n) \times 1$;
- 5) подає кожен вектор на вхід нейронної мережі;
- 6) визначає нейрон з максимальним значенням на виході;
- 7) інтерпретує індекс нейрона у цифру.

Для виводу індексу додаємо блок «InsertText». Остаточний вигляд моделі представлено на рис. 33.

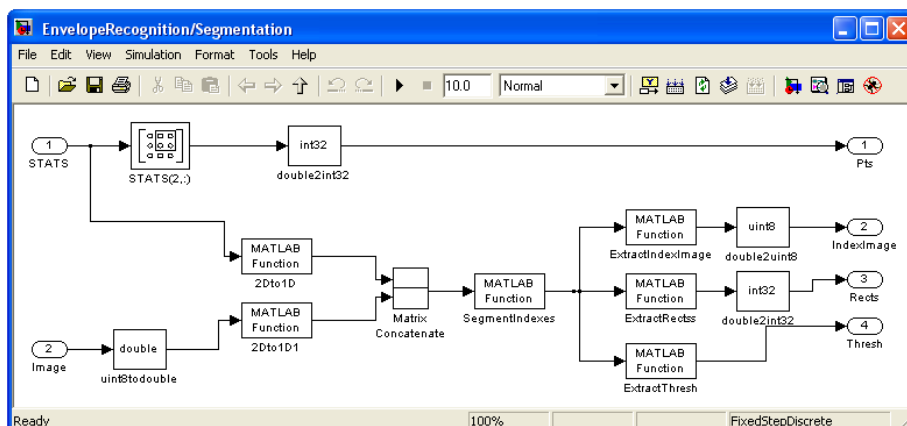
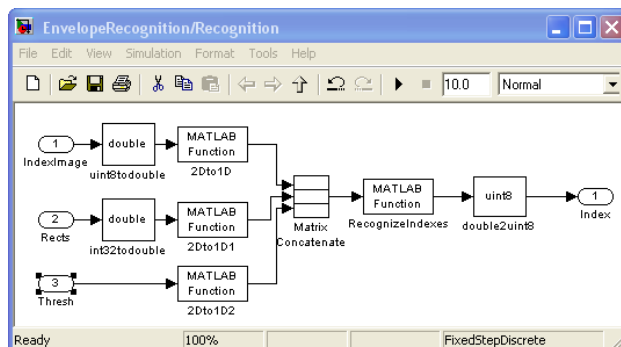
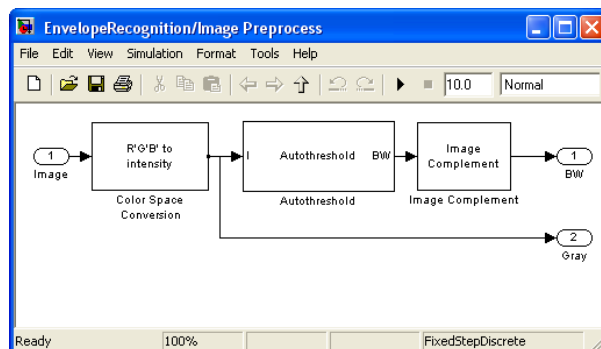
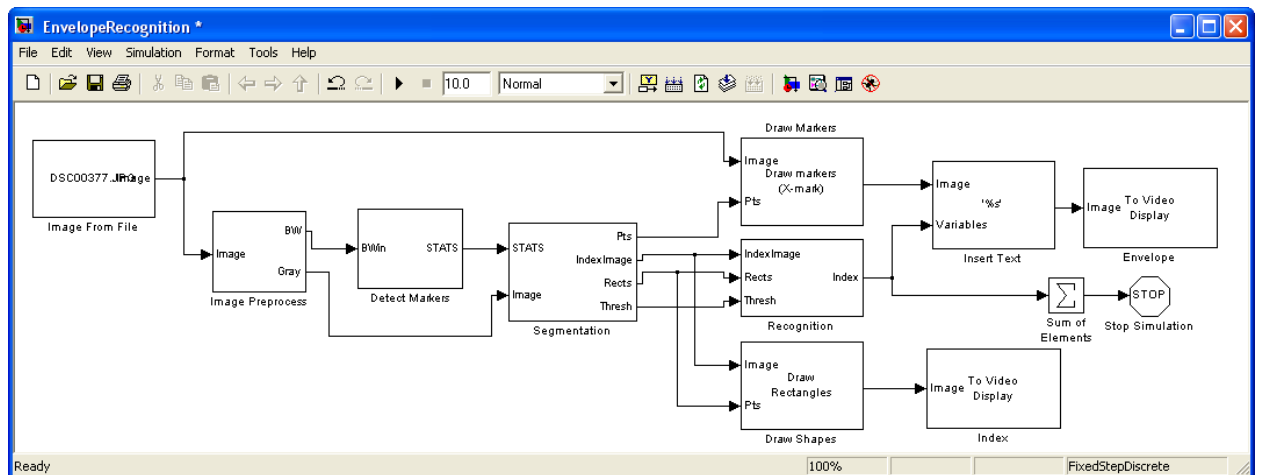


Рисунок 33 – Остаточний вигляд моделі «EnvelopeRecognition» в Simulink

При обробці зображення DSC00377.JPG модель видає результат, що показано на рисунку 34.

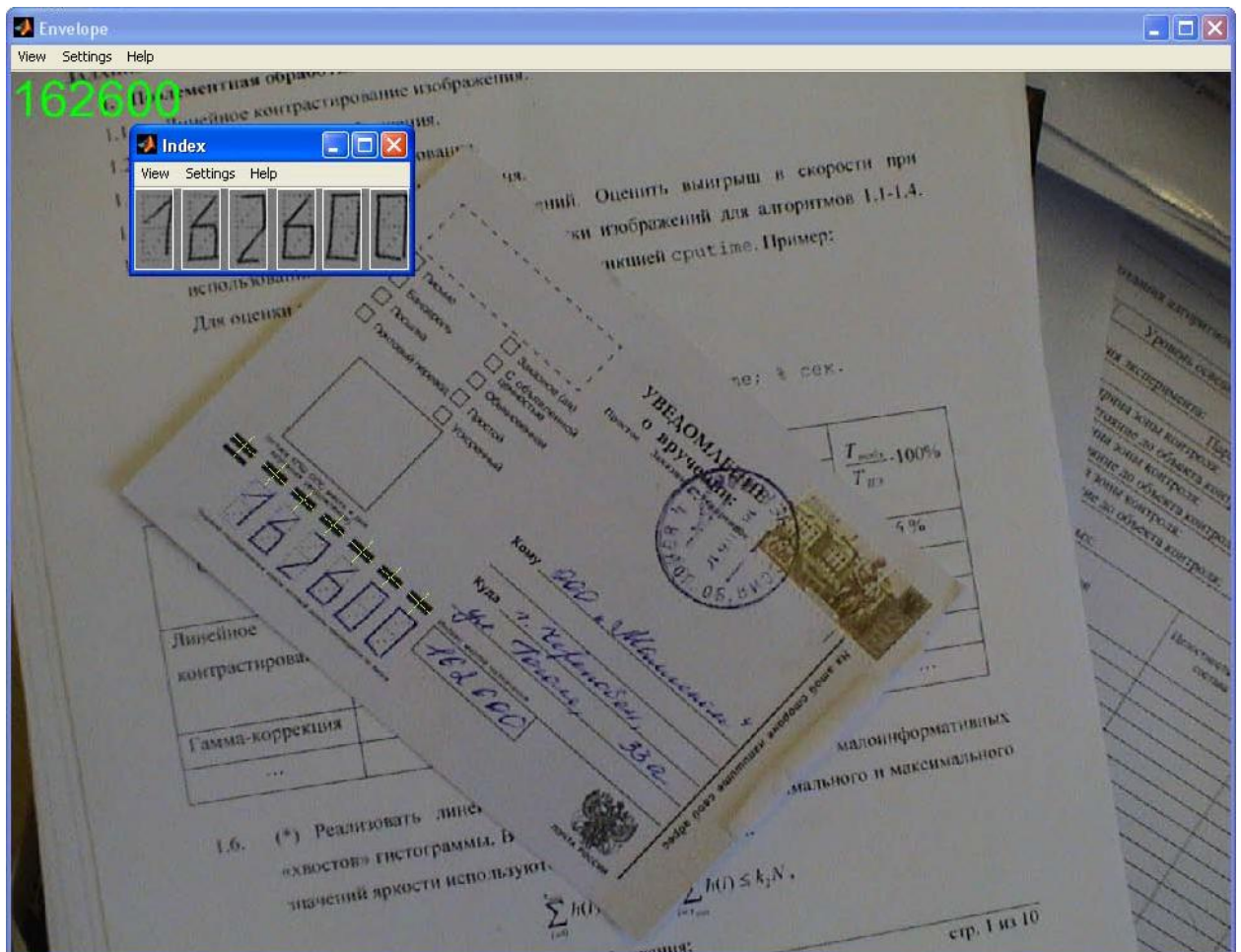


Рисунок 34 – Результат розпізнавання індексу на зображенні DSC00377.JPG

Отже, як можна бачити, модель працює та дозволяє коректно розпізнавати індекс на конверті.

ЗАВДАННЯ

Папка Model містить модель «EnvelopeRecognition» та функції у m-файлах:

extractindeximage.m,
extractrects.m,
extractthresh.m,
transform2Dto1D.m,
markerfilter.m,
segmentindexes.m,
recognizeindexes.m,

а також захищені p-файли:

markerfilterp.p,
segmentindexesp.p,
recognizeindexesp.p.

Наявність p-файлів необхідна для того, щоб впевнитися у працездатності алгоритму.

Запропонуйте власні реалізації функцій

markerfilter, segmentindexes та recognizeindexes!

Альтернативні завдання*:

- Розробіть власну систему оптичного розпізнавання символів (OCR), яка базуватиметься на розглянутих вище методах перетворення зображень у півтонове, бінаризоване, виконує сегментацію окремих символів та їх розпізнавання за допомогою попередньо навченої нейронної мережі.
- Розробіть власну систему детектування/розпізнавання одновірних, двовірних і QR штрих кодів. (Приклад реалізації є у нових версіях MATLAB Simulink).

ВИСНОВКИ

Результатом даної розрахунково-графічної роботи є імітаційна Simulink-модель, яка дозволяє виконувати розпізнавання індексу на конвертах. Створена модель передбачає розв'язання комплексної задачі технічного зору, яка в свою чергу складається з таких підзадач: перетворення кольорового зображення на півтонове; бінарізація півтонового зображення з автоматичним визначенням порогу; виділення компонент зв'язності; визначення маркерів індексу та фільтрація помилкових об'єктів; знаходження початкового та кінцевого маркеру; визначення кута нахилу; впорядкування розташування маркерів; поворот зображення; виділення зони індексу; сегментування зони індексу; розпізнавання зони індексу.

ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ

1. Яким чином виконується перетворення кольорового зображення у ахроматичне? В чому полягає особливість даного перетворення?
2. Як називається розглянутий в роботі метод бінаризації сірого (ахроматичного) зображення? В чому полягає його суть?
3. Дайте визначення компоненті зв'язності.
4. Яким чином в роботі виконується фільтрація маркерів індексу? Які для цього виконуються критерії? Якщо ви застосовували власні критерії опишіть їх.
5. Як виконується пошук компонент зв'язності?
6. Яким чином виконується поворот зображення?
7. Який метод для апроксимації яскравостей використовується при виконанні повороту зображення в даній роботі? Які ще методи апроксимації яскравостей при виконанні геометричних перетворень зображень вам відомі?
8. Наведіть модель штучного нейрона. Наведіть рівняння, за яким обчислюється його вихід.
9. Наведіть найбільш поширені функції активації штучних нейронів.
10. Назвіть відомі Вам архітектури штучних нейронних мереж. Який тип нейронної мережі використано в роботі?

ЛІТЕРАТУРА

1. Дзюба В. Г., Варфоломеев А. Ю. Системы технічного зору. Курс лекцій для студентів спеціальності «Радіоелектронні апарати та засоби». – 2011. – 148 с.
2. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2005. – 1072 с.
3. Гонсалес Р., Вудс Р, Эддинс С. Цифровая обработка изображений в среде Matlab. – М.: Техносфера, 2006. – 616 с.
4. Дьяконов В. П. MATLAB 6/ 6.1/ 6.5 + Simulink 4.5. Основы применения. Серия «Полное руководство пользователя». – М.: СОЛОН-Пресс, 2004. – 768 с.